Chapter Five: Contents

(Traffic Microsimulator – 14 July 2001 – LA-UR 00-1725 – TRANSIMS 2.1)

Chapter Five: Figures

Chapter Five: Tables

Chapter Five—Traffic Microsimulator

1. INTRODUCTION

1.1 Overview

The Traffic Microsimulator module simulates the movements and interactions of travelers in a metropolitan region's transportation system. Using a trip plan provided by the Route Planner, each traveler attempts to execute the plan on the transportation system. The combined traveler interactions produce emergent behaviors such as traffic congestion.

The Traffic Microsimulator simulates

- intermodal travel plans,
- multiple travelers per vehicle,
- multiple trips per traveler, and
- vehicles with different operating characteristics.

Emphasis was initially placed on roadway transportation because of its high use, complexity, and importance to air quality. The roadway network includes freeways, highways, streets, ramps, turn pocket lanes, and intersections (with and without traffic signals). Drivers executing trip plans accelerate, decelerate, turn, change lanes, pass, and respond to other vehicles and signals.

Using a cellular automata (CA) approach, the Traffic Microsimulator provides the computational speed necessary to simulate an entire region at the individual traveler level. The CA approach provides a means to simulate large numbers of vehicles and maintain a fast execution speed.

Each link in the transportation network is divided into a finite number of cells. At each timestep of the simulation, each cell is examined for a vehicle occupant. If a vehicle is present in the cell, the vehicle may be advanced to another cell using a simple rule set. To increase fidelity, we would decrease the cell size, thus adding vehicle attributes and expanding the rule set results in slower computational speed.

We evaluate the fidelity and performance limits of the Traffic Microsimulator to establish the computational detail that supports the fidelity necessary to meet analysis requirements.

The sheer number of travelers and the level of detail in the microsimulation require that we use multiple CPUs where available. The *Algorithm* section of this chapter provides an explanation of the information flows and scheduling required to support parallel computing. The following sections present an overview of the simulation as it could be carried out on a single CPU.

Step One A representation of the transportation network is read in. This representation is very similar to a detailed street map; it includes a number of lanes, turn pockets, merging lanes, turn signals, and so on. Vehicles traveling along streets in the road network are simulated in detail. In addition to the streets, there are several kinds of accessories that represent parking lots, activity locations, and transit stops, all of which act like buffers for travelers who are not in a vehicle traveling on a street.

Step Two Each vehicle’s type and initial location are read in. Once this is complete, each traveler’s plans are read in (as needed).

Step Three Travelers are placed on the network and are allowed to travel from their point of origin to their final destination. For non-simulated modes, this movement is simple—a traveler is removed from the buffer in one accessory and placed in the buffer on another, with a new departure time reflecting the trip’s estimated duration.

Vehicles move from one grid cell to another by using a modified CA approach. Modifications in this approach support lane changing and plan following for each vehicle until it reaches the end of a link. There the vehicles wait for an acceptable gap in traffic or for protection from a signal before they move through the intersection onto the next link. This continues until each vehicle reaches its destination, where it is removed from the network.

1.2 Traffic Microsimulator Major Input/Output

Fig. 1 provides an overview of the input and output involved with the Traffic Microsimulator. At a minimum, TRANSIMS Network information must include

- the location of streets and intersections,
- the number of lanes on the streets,
- the manner in which the lanes are connected,
- some parking locations on the streets, and
- a collection of activity locations.

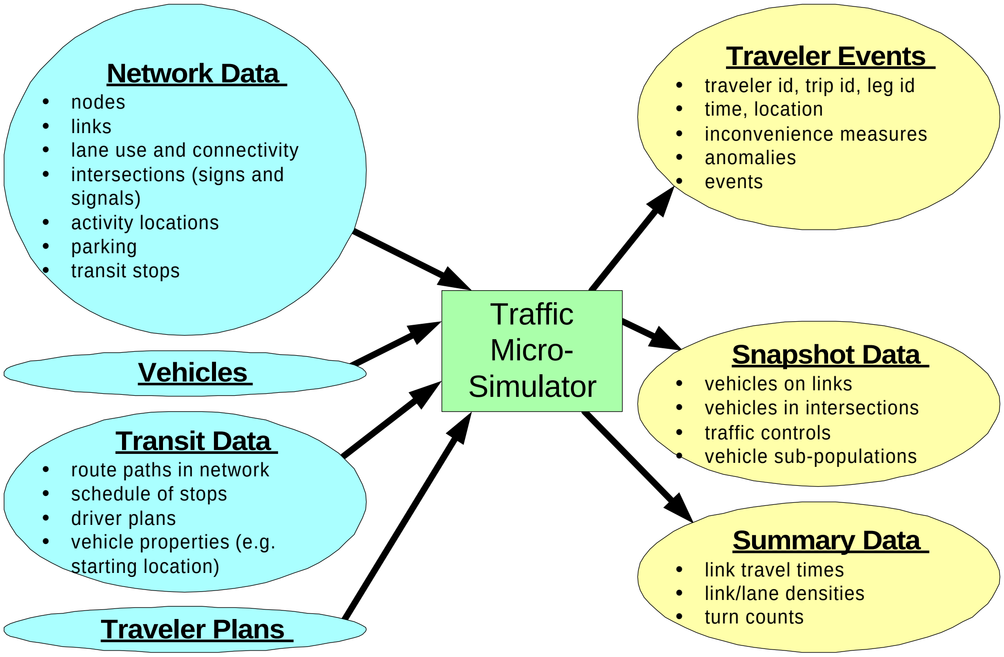

Fig. 1. This graphic shows what data go in and what data come out of the Traffic Microsimulator.

Some studies benefit from, or require more, detailed information about the network. For example, the Traffic Microsimulator is capable of using turn pockets and merge lanes, lane-use restrictions (such as high-occupancy-vehicle lanes), turn prohibitions, and speed limits. Each intersection has a controller (examples include a stop or yield sign, a traffic signal, or even a set of coordinated traffic signals).

Another type of beneficial network information consists of a list of transit stops serviced by each transit route. The actual transit schedule is encoded in the travel plans of transit drivers. Transit drivers stop to pick up or drop off passengers at transit stops.

The Traffic Microsimulator must have a complete description of each traveler’s transportation plans. A plan is broken down into a sequential set of trips, which must begin and end at an activity location (such as home, work, or shopping center). A trip is further decomposed into a set of unimodal legs. A traveler can use only a single mode of transportation on a leg. Accordingly, several legs are chained together to form a single trip.

2. TRAFFIC MICROSIMULATOR DESCRIPTION

2.1 Overview

As a simulated day progresses, each person follows a predefined plan to move from one activity to the next by using combinations of modes, such as walking, driving a vehicle, and riding in a (private or public) vehicle. The Route Planner provides a link-by-link travel plan of the traveler, including the mode of travel.

All TRANSIMS vehicles are simulated in sufficient detail to include driving on roads, stopping for signals, accelerating, decelerating, changing lanes, stopping to pick up passengers, and so on. Mode changes (e.g., from walking to car or to transit) are explicitly simulated based on information contained in the traveler's plan.

Vehicles follow a simple set of rules that guarantee no collisions will take place. Phenomena such as reaction times and limited visibility are not simulated explicitly. However, the effects of these phenomena are simulated by the values of parameters used in the driving rules so that the fundamental flow-density diagram matches real, observed traffic.

The simulation can estimate the impact of hypothetical changes on quality of service. It provides answers to questions such as the following:

- If a proposed highway were built, what would be its effects on traffic patterns?
- How would a change in transit schedules affect riders?
- Can changing an intersection's traffic signals alleviate congestion?
- Are there common demographic characteristics of the subpopulation most affected by a particular infrastructure change?

The Traffic Microsimulator's analytical power resides in its ability to aggregate the results of millions of interactions within the transportation system. The following sections focus on the level of detail used to simulate a travel plan.

2.2 Single-Trip Example

The following example consists of a six-leg multimodal work-to-home trip. It begins and ends at activity locations coded in the TRANSIMS networks.

Leg 1: walk from activity location *W* to bus stop *X*, where *W* is the work activity location and *X* is a bus stop in the network description.

Leg 2: take route *Y* to bus stop *Z*.

Leg 3: walk to parking lot *P*.

Leg 4: drive to day care at activity location *D*.

Leg 5: drive (with one passenger) to parking location *P2*.

Leg 6: walk to activity location *H* (home).

2.2.1 Walking Legs

For walking legs, TRANSIMS does not explicitly microsimulate the second-to-second locations of pedestrians. The traveler arrives at the destination at a simulation time computed by adding the delay time (contained in the plan) to the start time for the walk-mode leg. No additional information is required or generated for walk-mode legs.

2.2.2 Bus Legs

Bus-leg plans require one piece of additional information: the acceptable route. The precise itinerary of the bus the traveler gets on is determined by the driver's plan. The traveler simply boards the bus at a bus stop and rides it until his or her desired stop is reached, at which point he or she exits the bus.

The microsimulation explicitly represents bus loading and unloading. Resource constraints are observed, such as vehicle capacity and transit stop capacity. If a bus is full when it reaches the bus stop, a traveler is not permitted to board and will wait for the next bus on the same route. With this level of detail, it is easy to determine how many passengers cannot find space on the bus or how many minutes a traveler must wait for a bus.

2.2.3 Parking Lot

After getting off the bus, the traveler must walk to the parking lot. In this instance, the parking lot is where the traveler left his or her private vehicle. This walking leg is handled as previously described.

2.2.4 Driving Legs

Upon arriving at the parking lot, the traveler is associated with a specific vehicle, which either must have been left in the parking lot earlier in the simulation or placed there during initialization.

The traveler and car exit the parking lot and enter the traffic network. The traveler's plan specifies exactly which turns he or she will take until he or she arrives at the daycare center. At this point, the traveler waits until the passenger enters the vehicle. The passenger's plan will specify what vehicle to ride in, and the passenger will be waiting for this vehicle to arrive. The driver's plan specifies how many passengers to pick up. Once again, the driver re-enters the transportation network, completing the remainder of the planned trip.

2.2.5 Realistic Simulation

As described above, a transit schedule is implemented by providing plans for each transit driver. Like other travelers, a driver can switch vehicles, switch routes (with or without switching vehicles), and take layovers of prescribed duration or ending time at specific control points.

The Traffic Microsimulator enforces physical constraints—travelers cannot be in two places at once, and they cannot create vehicles. Information in the plan file initiates and places travelers in their initial start locations, whereas information in the vehicle file places vehicles in their initial locations.

2.3 Cellular Automata

Cellular automata simulation yields vehicle movement. Each roadway section is divided into grid cells, each of which is one lane wide and 7.5 meters long (see Fig. 2). Each cell contains either a vehicle (or a part of one) or is empty.

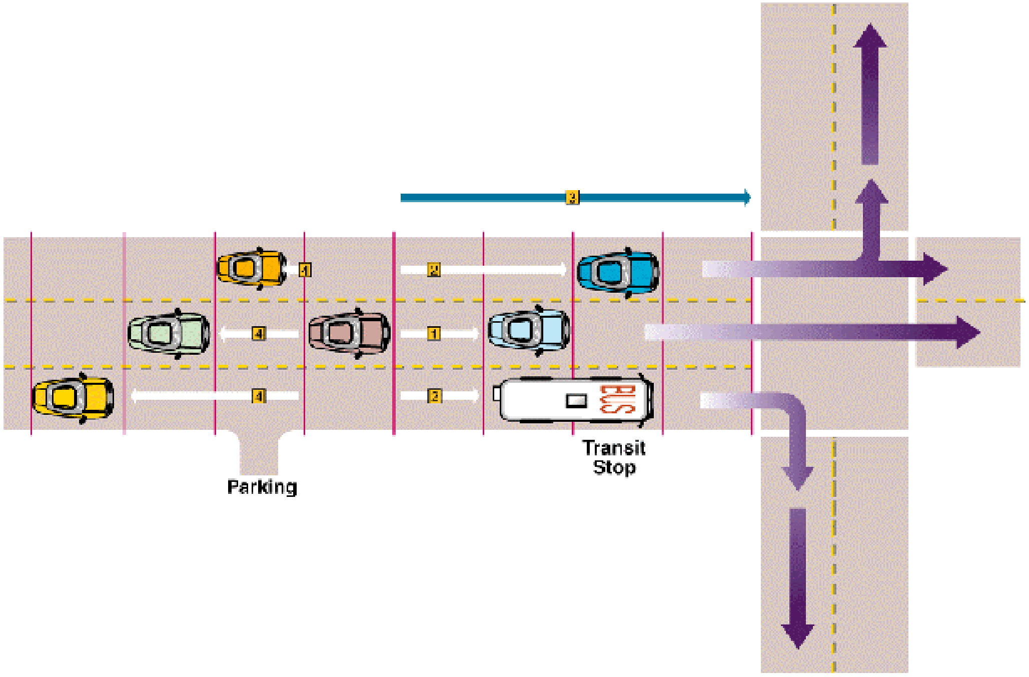

Fig. 2. Shown from the perspective of the middle peach-colored car, the gap (labeled "1" in the figure) between vehicles determines accelerations and influences lane changes. The gaps (2 and 4 for left-lane change, 6 for right-lane change) to vehicles in other lanes also influence lane changing. The distance to the intersection (3) determines the relative importance of changing lanes. The gaps (4, 5, and 6) to upstream vehicles from a parking facility determine whether vehicles can exit the parking facility.

The simulation is carried out in discrete timesteps, each simulating one second of real time. On each timestep, a vehicle on the network decides whether to accelerate, brake, or change lanes in response to the occupancy of nearby grid cells. After every vehicle is allowed to make these decisions, they are all moved to new grid cells in accordance with their current velocity.

As part of the lane-changing procedure, transit vehicles scan nearby cells for transit stops, which they must service. The transit vehicles examine the queue at the stop to see if anyone is waiting for the vehicle; they also query their passengers to see if anyone wants to get out at the stop. Finally, a transit driver's plan may specify a departure time from any stop—the driver must enter the stop if the scheduled departure time has not yet been reached.

If the vehicle must stop, it either stops in the cell next to the transit stop or pulls off the grid (depending on the type of transit stop). Passengers take a fixed time to enter or leave the vehicle.

Each intersection has traffic-control logic that directs the entry of vehicles into the intersection. Traffic controllers examine the traffic in each lane at the intersection. If the intersection is clear, vehicles pass through it in a fixed amount of time and are placed on the next roadway's link.

Vehicles entering the roadway from parking locations or off-street transit stops can enter any lane with a large enough gap between it and oncoming traffic. The gap must be wide enough to ensure that, on the next timestep, no vehicles collide with vehicles entering the roadway.

The simulation guarantees that each vehicle makes decisions based on the state of every other vehicle in its local vicinity (i.e., five cells) at the same time. In other words, every vehicle on the network makes its acceleration decision based only on information available at time t, which does not include the time $t+1$ positions of vehicles that already have made their acceleration decision. This parallel update scheme ensures that the simulation results do not depend on the order in which streets in the network are updated.

To accomplish a simulation update, a single timestep is broken down into several steps (see Fig. 3).

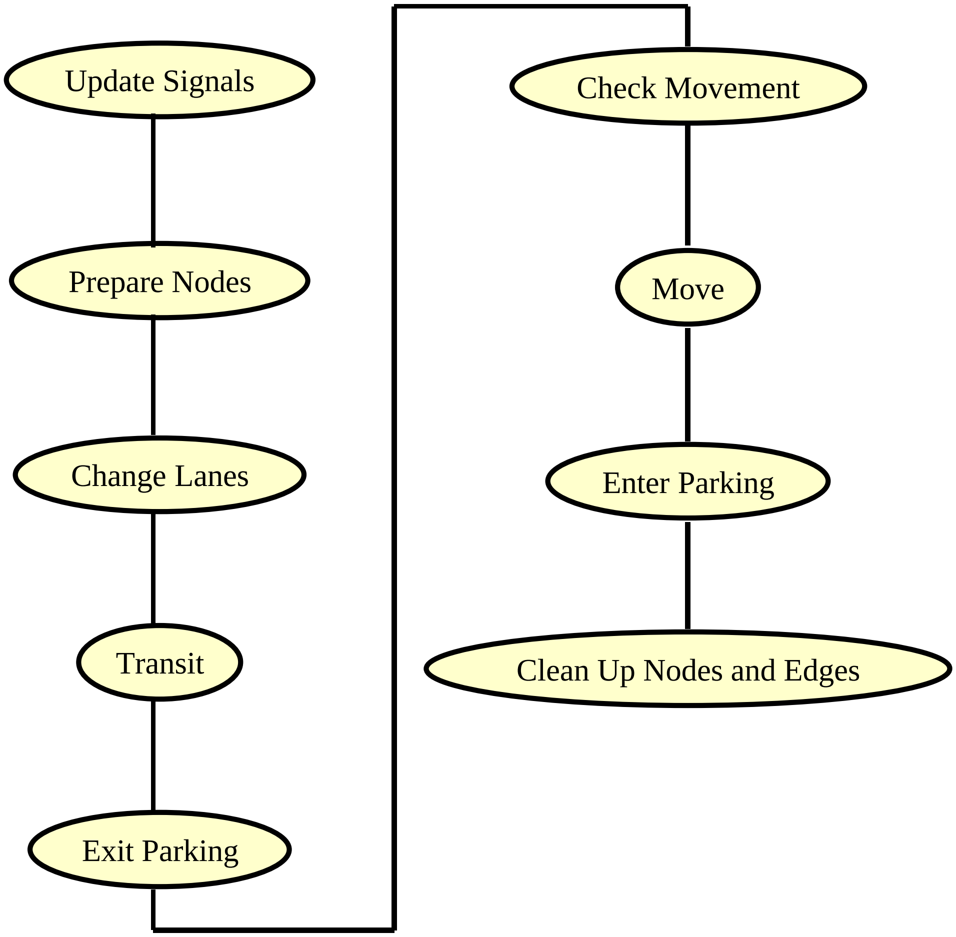

Fig. 3. This figure shows the order of execution of processes involving vehicles in each timestep.

Step One Update Signals

- Update traffic signals.

Step Two Prepare Nodes

- Vehicles in the intersection reserve space on their destination link (if possible).

Step Three Change Lanes

- All vehicles are allowed to change lanes. In this step, transit vehicles are also allowed to enter transit stops.
- To avoid possible collisions when vehicles in two lanes at time *t* both try to change into the same lane, we alternate the direction of lane changing every timestep.

Step Four Transit

- Allow transit vehicles to exit transit stops.
- Allow vehicles stopped at transit stops to collect and disembark passengers.

Step Five Exit Parking

- Vehicles at the head of a queue waiting to leave a parking location are allowed to enter the road.

Step Six Check Movement

- Those vehicles entering intersections are marked and given instructions from the intersection controller about the availability of their destination link.

Step Seven Move

- Every vehicle on a grid makes its acceleration decision; all of the vehicles are moved.

Step Eight Enter Parking

- Vehicles are allowed to exit into parking locations.

Step Nine Clean up Nodes and Edges

- Vehicles leave intersections and appear in the space reserved for them by `Prepare Nodes`. Various temporary markers are removed from the grid cells.

2.4 Traffic Microsimulator Output

The Traffic Microsimulator produces four major kinds of output. Fig. 4 shows these four types:

- summary data (spatial and temporal),
- traveler events, and
- snapshot data.

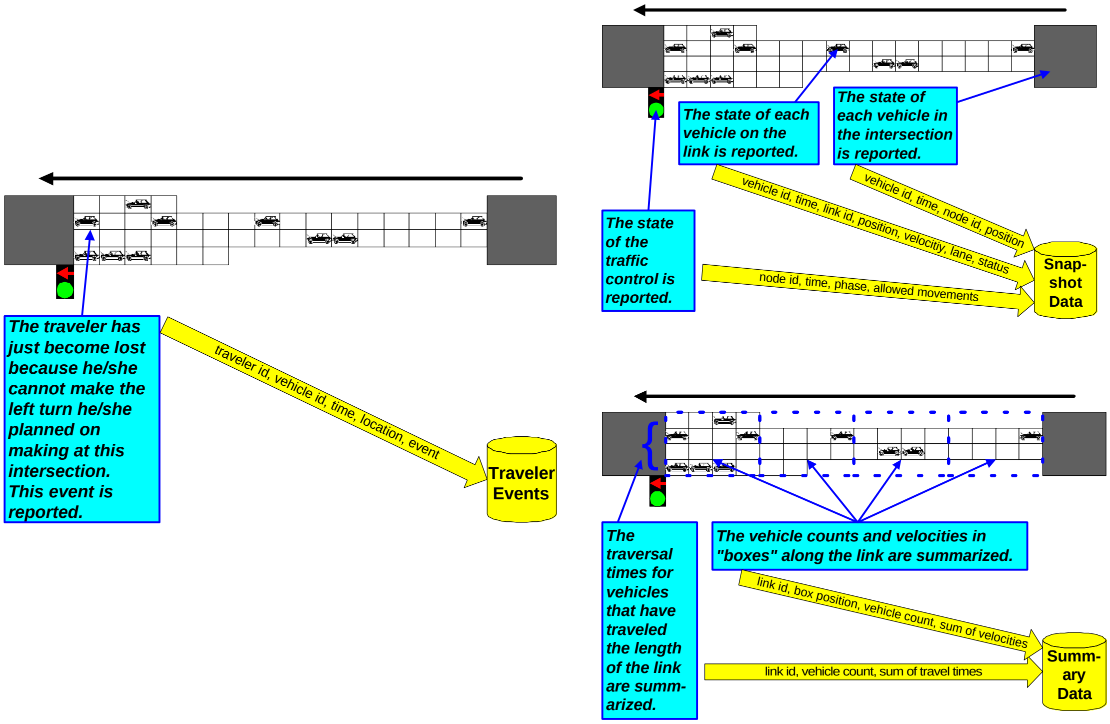
The state of each vehicle on the link is reported.
The state of each vehicle in the intersection is reported.
vehicle id, time, node id, position
vehicle id, time, link id, position, velocitiy, lane, status
The state of the traffic control is reported.
node id, time, phase, allowed movements
Snap-shot Data
The traveler has just become lost because he/she cannot make the left turn he/she planned on making at this intersection. This event is reported.
traveler id, vehicle id, time, location, event
Traveler Events
The traversal times for vehicles that have traveled the length of the link are summ-arized.
The vehicle counts and velocities in "boxes" along the link are summarized.
link id, box position, vehicle count, sum of velocities
link id, vehicle count, sum of travel times
Summ-ary Data

Fig. 4. The various types of Traffic Microsimulator output. Depicted in this figure are traveler events, snapshot data, and summary data.

Spatial summaries include data aggregated over user-defined sections of roadway defined along street networks (e.g., densities and total flow in a 150-meter section).

Temporal summaries include data about travel times along streets at various times of day. Almost anything that happens to a traveler can be reported as a time-stamped event in the event output. Commonly used events include begin/end waiting at a given location (such as a bus stop), begin/end a leg, pass through an intersection, and enter a vehicle.

TRANSIMS can produce traffic animation from snapshot files (if desired), which contain time, position, and velocity information for each vehicle in the simulation. These files can be also used to recover data that have not already been provided in the summary data files. For instance, if a new study requires the average gap between vehicles, it can be computed from snapshot data.

Dumping out the snapshot data for a 24-hour simulation of a major metropolitan area creates extremely large files. Users are allowed to restrict output to smaller portions of the network and specific times during the simulation, as well as to select only a few desired fields or only those records that meet certain criteria. For example, a user may choose only specific events (such as beginning a leg), particular travelers, or vehicles traveling above a given speed. The sampling rate and reporting frequency for each data type are controlled by user-selected parameters. For more information, see the Output System documentation.

3. ALGORITHM

3.1 Overview

The procedures invoked during each simulation timestep can be placed into one of five broad categories:

1) placing travelers and vehicles,
2) updating the location of each traveler and vehicle,
3) preparing for a timestep,
4) cleaning up after a timestep, and
5) supporting parallel computation.

The following sections address these five procedures.

3.2 Placing Travelers and Vehicles

Fig. 5 shows the processes and data structures involved in loading travelers and vehicles into the Traffic Microsimulator.

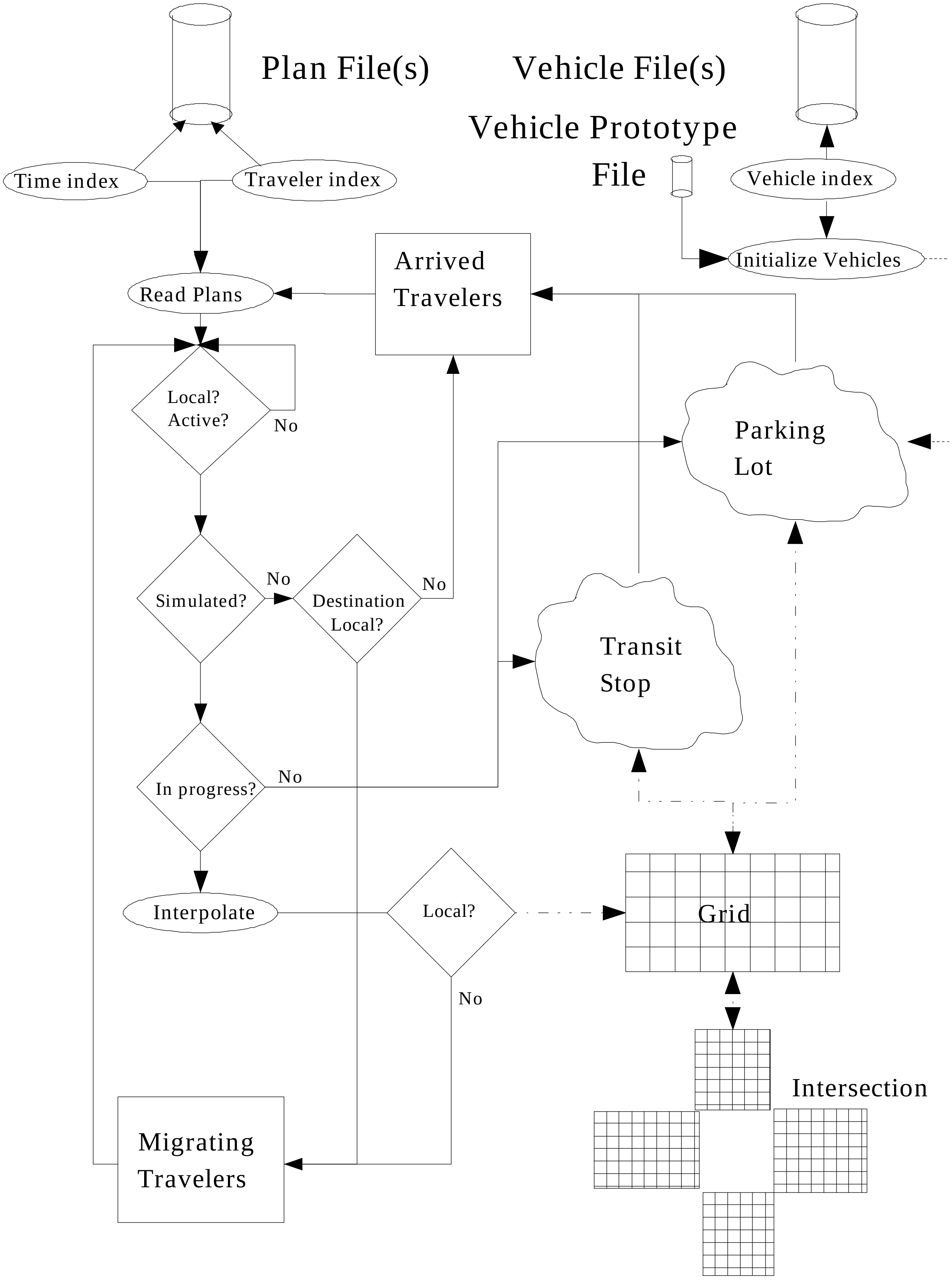

Fig. 5. A flow chart of the processes and data structures involved in loading vehicles and travelers into the Traffic Microsimulator. Vehicles flow through the dotted lines; travelers flow through the solid lines, and both flow through the dot-dashed lines.

The vehicle, vehicle prototype, and transit route files contain all of the information required by the Traffic Microsimulator in addition to the network files (not shown in Fig. 4). Vehicle and plan files are accessed through an index, which will be generated from the appropriate file if it does not already exist. Note that an index can refer to more than one data file. Furthermore, there may be a separate index for each processor (if the configuration file key `CA_USE_PARTITIONED_ROUTE_FILES` is set).

Traveler plans (i.e., legs of a plan) are read using the index sorted by expected departure time until all plans departing before or on the current simulation step have been read. In addition, the IDs of "hibernating" travelers (those who have already executed one leg of their plan and are waiting to depart on another) are popped off the queue of Arrived Travelers.

Each hibernating traveler carries along a minimal required information set that consists of

- traveler ID,
- current trip and leg ID, and
- a set of state flags used in maintaining states required by the output system.

To minimize memory requirements, other non-essential information is deleted from memory while a traveler hibernates. To find the next leg for each of the arrived travelers, the *Read Plans* process uses an index into the plan file that is sorted by traveler ID. Each plan must pass two tests before the traveler is placed onto the transportation network:

1) It must be "local," meaning that its origin must be an accessory that is a part of the network under the control of the CPU. If the simulation uses only one slave (`CA_SLAVES` is set to 1), everything is local.

2) It must be "active," meaning that (1) its expected arrival time must be after the simulation start time, and (2) its departure time must be before simulation end time. The following configuration file keys define the simulation start and end times: `CA_SIM_START_HOUR, CA_SIM_START_MINUTE, CA_SIM_START_SECOND` and `CA_SIM_STEPS`.

If the plan calls for a non-simulated mode of travel (activity, walk, or bicycle) and the destination is local, the process places the traveler in the Arrived Traveler queue with a departure time specified by the plan. For example, the plan may say to use the later of a 10-minute duration or a specific time (e.g., 8:10 a.m.).

The simulation will add ten minutes to the current simulation time, compare it to 8:10 a.m., and place the traveler into the queue with a departure time equal to the later of the two. If the destination is not local, the traveler must migrate to another CPU, where he or she will be placed into the Arrived Traveler queue for that CPU.

If the traveler is using a simulated mode of transportation (anything involving a vehicle), either as a passenger or a driver, and the plan is not in progress (i.e., its departure and arrival times do not straddle simulation start time), the traveler is placed in a queue in his or her origin accessory. This could be either a transit stop or a parking location. No travelers will be placed in activity locations because the simulated transportation modes do not have paths to or from such locations.

It is desirable for the simulation to reach normal traffic flow conditions as rapidly as possible. To facilitate this, vehicles whose driver's plans are in progress are placed on the roadway when the Traffic Microsimulator is initialized. This is based on where the driver's plans predict they will be at the simulation starting time.

Once a plan's geometric length is estimated, a link is selected by interpolating along the path according to the duration of the leg (as estimated by the Route Planner). The length is difficult to determine if the plan is not wholly contained within the part of the network local to the CPU, so this process is not guaranteed to produce the same initial condition when the number of CPUs varies.

If the interpolation process determines that a traveler should be placed on a non-local section of the network, it will add the traveler to the list of migrating travelers. Otherwise, the cell position and lane are randomly selected.

If the selected cell is already occupied, the grid is searched upstream for an available cell. If all cells upstream are occupied, the grid is searched downstream for an unoccupied cell. If all cells on the link are occupied, a warning message is printed and the vehicle is deleted.

No attempt is made to find an available cell on an adjacent link. Because interactions between vehicles are not taken into account, this procedure does not produce the same distribution of traffic that would be found by starting earlier and letting the simulation evolve to the same time.

Furthermore, transit passengers are not placed in transit vehicles, but rather they are placed at their destinations. If this interpolation scheme does not work satisfactorily, the user should start the simulation at an earlier time.

3.3 Updating Traveler Locations

After reading in and placing travelers, the Traffic Microsimulator executes their plans one step at a time. Each step involves several substeps in the order given in Fig. 3. The major data structures involved and their interactions during a timestep are sketched in Fig. 6.

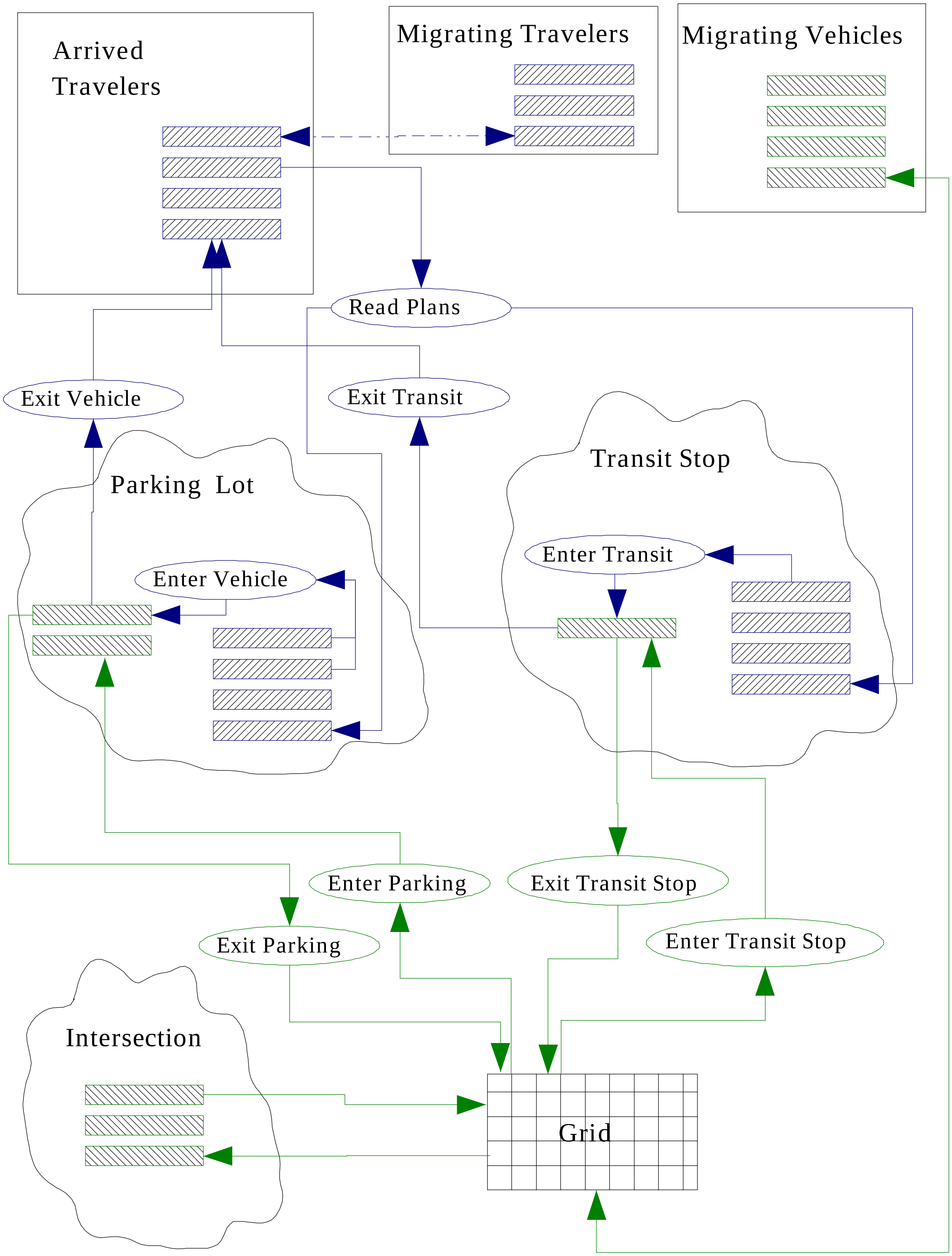

Fig. 6. This figure shows interactions among objects during a timestep. Travelers flow along blue lines; vehicles with travelers along green lines.

3.4 Traffic Dynamics

Interactions of individual vehicles on the transportation network produce traffic dynamics in the Traffic Microsimulator. To determine the position of vehicles on the roadway, a rule set is applied that governs movement and lane changes. This rule set must be as simple as possible to maintain the computational speed necessary for updating positions of the large number of vehicles that could be present in a regional traffic microsimulation.

The rule set imposes a no-collision strategy on the vehicles. Vehicle interactions based on the rule set combine to produce emergent driver behavior. Traffic dynamics require that, for any vehicle *v* at time *t*, all position change calculations must be based on other vehicle positions at time *t*, not at time *t+1*.

3.4.1 Lane Changes

During the timestep, we examine each vehicle and determine if it will change lanes. To produce realistic traffic dynamics, lane change and movement must take place on the same timestep.

Left and right lane changes are made on alternating timesteps to prevent collisions and to ensure that gap calculations are based on vehicle positions at time *t*, not *t+1*. Multilane roadways are processed from left to right when making left lane changes and from right to left when making right lane changes. Vehicles are not allowed to change into a lane if it would violate lane use or HOV restrictions.

A vehicle changes lanes for two reasons:

1) to pass a slower vehicle in the current lane, and

2) to make turns at intersections to follow its plan.

A vehicle that needs to make a turn at the next intersection (as part of its plan) will consider changing lanes when it is within a set distance from the intersection. As the vehicle approaches the intersection, the urgency to change into a lane increases linearly as the vehicle approaches the intersection (configuration file key `CA_PLAN_FOLLOWING_CELLS`). Any vehicles that fail to make the required lane changes are marked as off-plan.

3.4.2 Passing Lane Change

Passing lane changes are based on three gap calculations (see Fig. 2):

1) gap in the current lane (G_c),

2) gap forward in the new lane (G_f), and

3) gap backward in the new lane (G_b).

If these gaps satisfy the following constraints, a lane change will be attempted with probability `CA_LANE_CHANGE_PROBABILITY`:

1) $V + 1 > G_c$ (i.e., a vehicle ahead in the current lane is preventing acceleration)

2) $G_f > G_c$ (i.e., the gap in the neighboring lane is larger than in the current lane)

3) V £ G_f (i.e., the gap in the neighboring lane is large enough to maintain the vehicle’s current speed)

4) G_b ³ $V_{GlobalMax}$ (i.e., if the lane change were made, there would not be a collision).

$V_{GlobalMax}$ is used in constraint three rather than the actual speed of the other vehicle for efficiency. Nothing in the lane-changing or CA rules depends on the velocity of any vehicle besides the one under consideration.

3.4.3 Plan-following Lane Change

Acceptable approach lanes that allow a vehicle to transition to the next link in its plan are determined when a vehicle enters a link. A preferred lane is also selected. The preferred lane may change as the vehicle changes lanes. Plan-following considerations are introduced into lane-change calculations when a vehicle is within a set distance from an intersection (D_{PF}). The bias to make a lane change increases as the vehicle nears the intersection. The bias also increases linearly with the number of lanes away from an acceptable lane.

If the vehicle is already in an acceptable approach lane, the vehicle is biased to stay in the correct lane and ignore lane changes to pass slower vehicles (i.e., lane changes based on gaps).

Lane changes are controlled by introducing an additional parameter to the lane-change calculations. This parameter, W, is initially set to zero.

If a vehicle is within the D_{PF} but is not in an acceptable approach lane, W is set based on the distance between the vehicle and the intersection (D_I), and the minimum number of lane changes n it will take to reach an acceptable lane,

$$W = V_{Max} - \frac{(V_{Max} - 1)}{n \cdot D_{PF}} \cdot D_I$$

Note that as D_I goes from n ᵡ D_{PF} to 0, W goes from 1 to V_{Max}. The parameter W is used to gradually relax constraints 3 and 4. When W reaches V, constraint 3 is completely removed; when W reaches V_{Max} , constraint 4 is removed.

Because only one type of lane change is made during a timestep, the type of lane change needed (left/right) must be the same as the type of lane change (left/right) calculated during this timestep.

It is possible for a vehicle to have more than one approach lane that is acceptable for plan-following. If the vehicle is in an acceptable lane and the new lane (left/right) is also an acceptable approach lane, *W* = 0, which allows lane changes based on gaps. If the new lane is not acceptable, no lane change is allowed, unless the vehicle must cross the new lane to reach an acceptable one.

3.4.4 Special Cases

This section outlines several special cases involving lane changes.

3.4.4.1 Mass Transit

The algorithm handles mass transit vehicles separately because they must not become off-plan and because they must have priority in making lane changes. In addition, mass transit vehicles are allowed to enter transit stops during lane changes.

Each mass transit vehicle enters a transit stop if

- it is not full and there is a queue of people waiting at the stop,
- any passenger wishes to get off at the stop, or
- the driver's plan includes a scheduled departure time for this step and that time has not yet passed.

The vehicle either will be left occupying the grid cells or taken off the grid entirely, depending on the style of transit stop (e.g., `STATION` or `STOP`). If it is left on the grid, it will attempt to get into the rightmost lane. The vehicle's speed constraint is set to 0 while it is in the `STOP` style.

3.4.4.2 Merge Lanes

Merging is handled by using the lane-change logic. Vehicles in merge lanes are forced to make lane changes in the same direction as the merge direction. In some cases, a lane can have a merge pocket and a turn pocket further down the lane toward the intersection. In these cases, vehicles are prohibited from entering the lane until they are past the end point of the merge pocket.

3.4.4.3 Turn Pocket Lanes

The Traffic Microsimulator imposes speed restrictions on vehicles attempting to enter a turn pocket lane from an adjacent lane. These restrictions prevent movement of the vehicle past the start of the turn pocket, thus causing the vehicles to queue on the adjacent lane until it is a possible to execute a lane change into the turn pocket lane.

In Fig. 7, the vehicle in Lane 2 needs to make a left turn at the next intersection. The left turn pocket (Lane 1) has no vacant cells. At time *t*, the vehicle's speed is 3, which will move the vehicle past the start of the turn pocket. The vehicle's speed is constrained to 2 (the distance from the vehicle's current position at time *t* and the start of the turn pocket).

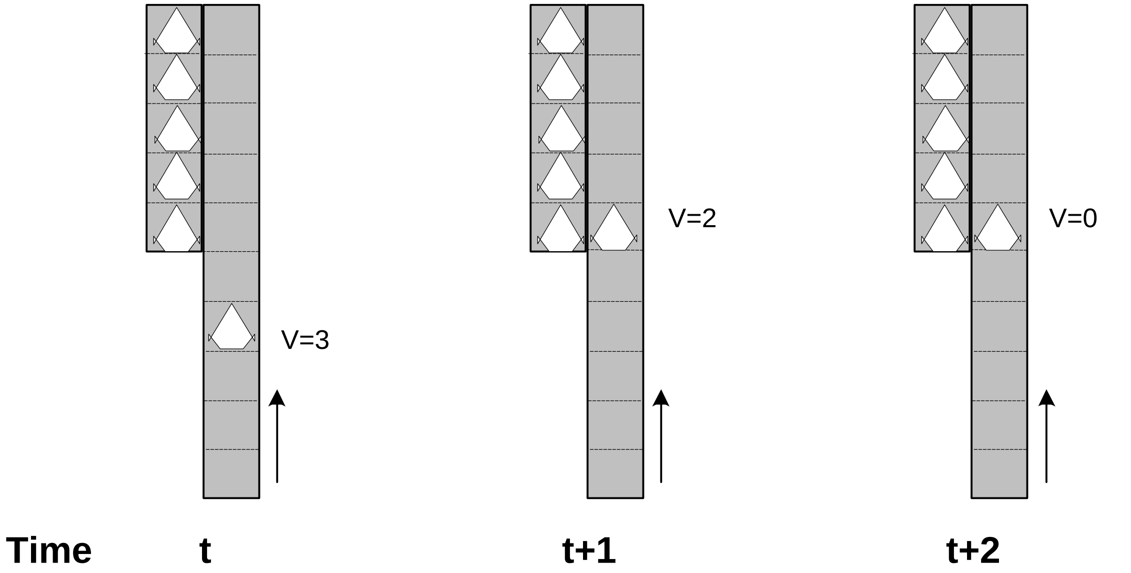

Fig. 7. This graphic shows vehicle behavior at turn pocket lanes.

At time *t+1*, the vehicle has moved down Lane 2 to the starting cell of the turn pocket. A lane change into the turn pocket is not possible because other vehicles occupy all of the cells. By constraining the speed to 0, the vehicle is prevented from traveling further down Lane 2. At time *t+2*, the vehicle remains in Lane 2 with speed 0. The vehicle's speed will remain constrained to 0 until a lane change into the turn pocket is possible.

3.4.4.4 Look Ahead Across Links

Approach lanes can be determined by considering only the connectivity to the next link. However, some vehicles cannot make the required lane changes into acceptable approach lanes on short multilane links with multiple lane connectivity at the intersections. Looking ahead across links increases the time that a vehicle has to make a plan-following lane change.

The Traffic Microsimulator uses plan look-ahead distance (set by the configuration file key `CA_LOOK_AHEAD_CELLS`) to determine acceptable approach lanes. The distance is used to determine how many links in the plan are considered when determining the approach lanes on the current link.

- A default value of 0.0 means that approach lanes are determined by considering the next link only.

3.5 Transit

While a vehicle is in a transit stop, the transit-stop object contains a pointer to the vehicle (implying that the capacity of stops is 1). The object also contains queues of travelers waiting to board transit vehicles. There is a separate queue for each route ID.

If there is a transit vehicle currently servicing the stop and it has been there for at least the number of timesteps specified by the configuration file key `CA_TRANSIT_INITIAL_WAIT`, travelers are allowed to enter and exit the vehicle. Entry and exit can take place simultaneously, but the mean rate at which travelers enter and exit is set by the configuration file keys `CA_ENTER_TRANSIT_DELAY` and `CA_EXIT_TRANSIT_DELAY`, respectively.

Travelers are popped off the traveler queue until it reaches either the maximum number of travelers who can board in a single timestep or a traveler whose next departure time is later than the current simulation time.

If a traveler’s plan calls for him or her to take the route that this vehicle is servicing, and the number of passengers already aboard does not exceed the capacity for this type of vehicle specified by the vehicle prototype file, he or she will enter the vehicle.

Travelers leaving the vehicle have completed a leg of their plan; they are placed in the *Arrived Travelers* list to trigger the *Read Plans* process to find the next leg of their plans. If all of the passengers exiting at this stop have been taken care of and either the bus is full or no more passengers are waiting to board, the vehicle is placed back on the grid (if necessary) and its speed constraint removed.

3.6 Exiting from Parking Places

A parking place accessory has a list of IDs for the vehicles present (either because they began the simulation there or they have arrived during the course of the simulation). It also has a queue of travelers and their associated plans. This procedure handles each traveler in the traveler queue whose departure time has arrived.

A vehicle whose ID is on the list will have been instantiated in the simulation only if it has arrived here from somewhere else. Otherwise, a new vehicle with this ID must be created using the type implied by the traveler’s plan.

A traveler cannot leave unless his or her vehicle is present. If the vehicle is not there, the traveler’s departure time is incremented and he or she is replaced in the queue.

Depending on his or her plan, the traveler is added to the vehicle as either a driver or a passenger. If the driver has not yet been added to the vehicle, the next traveler is popped off the queue. Otherwise, the driver determines how many passengers are anticipated. (This information is contained in the driver’s plan, along with the IDs of the expected passengers.)

If any passengers are missing, the driver is placed back in the queue so that the vehicle will try to leave again on the next timestep. If the driver and all passengers are present, the vehicle attempts to find room on the grid in any lane, not in the boundary region (as described in “Distributed Links and Boundaries Information Flow”), traveling at the speed limit.

Once the appropriate grid for the planned direction of travel is determined, the grid is searched upstream for a distance of V_{Max} cells. If a vehicle is found in a lane, that lane and the adjacent lanes are eliminated from consideration. Thus, the maximum number of vehicles that can leave a lot in one timestep is the number of lanes on the grid.

All lanes are searched and if a lane is available, the vehicle is placed on the lane at the cell corresponding to the parking place location. If there is no room on the grid, the driver is returned to the traveler queue.

3.7 Movement Check/Intersections

This procedure handles vehicles that leave a link and pass through an intersection. Upon arriving at an intersection, a vehicle’s destination lane on the next link is determined. The current lane is selected if it is allowed on the next link; otherwise, a lane is picked at random from the set of allowed lanes. This set takes into consideration lane use and HOV restrictions.

Unsignalized intersections with stop/yield traffic controls require vehicles to consider oncoming traffic before they can move onto the next link. The vehicles use the gap between the oncoming vehicles and the intersection to determine if they can enter the intersection. If the gap is acceptable, the vehicle traverses the intersection and arrives on the destination link during a single update step in the microsimulation.

Vehicles at signalized intersections have different behaviors from those at unsignalized intersections. When a vehicle enters an intersection, it is placed in a queued buffer, where it resides for a specified time before exiting to the destination link. The time that the vehicle spends in the queued buffer models the time necessary to traverse the intersection. Vehicles with permitted but not protected movements from the intersection traffic control must consider the oncoming traffic before entering the intersection. The configuration file key for this is `CA_INTERSECTION_WAIT_TIME`.

To enter an intersection, a vehicle must satisfy six conditions, all of which are outlined below.

Condition One Be the first vehicle on the link in the current lane going toward the intersection. Only one vehicle per lane is allowed to enter the intersection in a single timestep.

Condition Two Have a current speed greater than or equal to the number of empty cells between the vehicle and the end of the link.

Condition Three Satisfy the conditions of the traffic control at the intersection. The state of the traffic control indicates if a vehicle must consider oncoming traffic gaps.

Condition Four Ensure that there is an acceptable gap between the vehicle and oncoming traffic.

Condition Five Ensure that the intersection buffer for the current lane is not full.

Condition Six Ensure that the destination cell in the destination lane on the destination link is unoccupied.

A vehicle will attempt to enter an intersection if its current speed is greater than or equal to the number of empty cells between the vehicle and the end of the link. The state of the traffic control at the intersection is an important factor in determining if a vehicle can enter the intersection.

To enter a signalized intersection, the traffic controller must indicate a permitted, protected, or caution movement for the current lane and the desired movement. At an unsignalized intersection, stop and yield signs impose conditions on intersection entry.

The traffic controller state may require that the distance between the intersection and oncoming traffic (interfering lane gap) meet certain criteria before the vehicle can enter the intersection. Table 1 shows the traffic controller states and their corresponding actions.

Table 1. Traffic controller states and corresponding actions.

Traffic Controller State	Action	Conditions
S* - Protected	Proceed	None
S – Wait	Stop	None
S – Permitted	Evaluate	G_I on IL (Interfering Lanes)
S – Caution	Proceed	None
U** -None	Proceed	None
U – Stop	Wait Evaluate	Stopped < 1 Timestep G_I on IL, Stopped ≥ 1 Timestep
U – Yield	Evaluate	G_I on IL

* S = Signalized intersection
** U = Unsignalized intersection

The interfering lane gap (G_I) consists of the distance between the oncoming vehicle and the intersection. The oncoming vehicle must be on a link connected to the intersection, which limits the look-back distance for oncoming traffic to the length of a single link.

The oncoming vehicle's speed (V_{OV}) and the Gap Velocity Factor (GVF), specified by the configuration file key `CA_GAP_VELOCITY_FACTOR`) are used to calculate the Desired Gap.

Desired Gap (G_d) = V_{OV}*GVF

On links in which the desired gap is greater than the number of cells on the link, the number of cells on the link is used as the desired gap.

$G_I \geq G_d$, Interfering Gap Acceptable

$G_I < G_d$, Interfering Gap Not Acceptable

Note that for an oncoming vehicle with speed of 0, G_d will be 0, which allows movement through intersections in congested conditions in which both G_d and $G_I = 0$. If the interfering gap is not acceptable, the vehicle is at a stop or a yield sign, and the interfering lane is also controlled by a stop/yield sign, then there will be a deadlock resolution in which the vehicle will proceed with probability determined by the value of the configuration file key `CA_IGNORE_GAP_PROBABILITY`.

As shown in Fig. 8, a vehicle can enter the intersection only when the interfering gaps are acceptable ($G_I \geq G_d$).

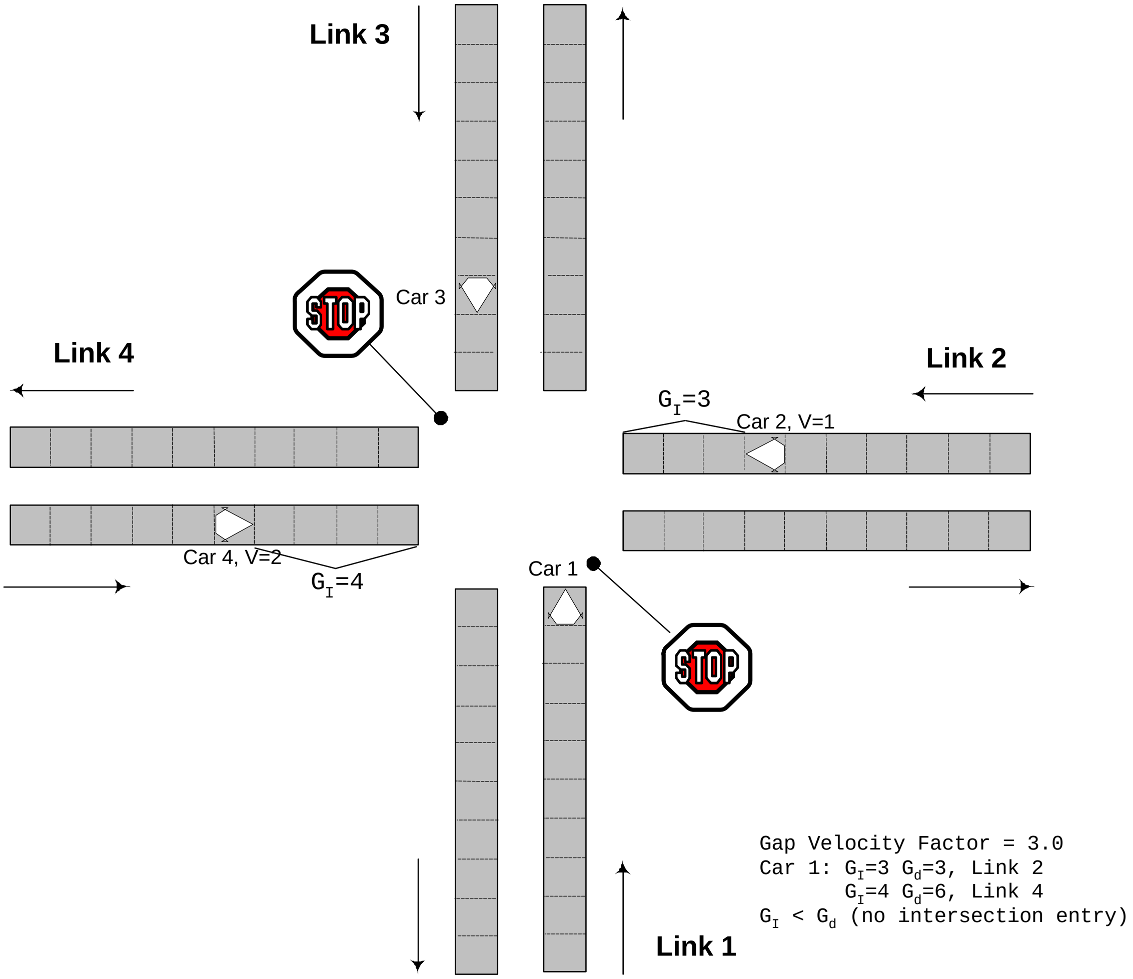

Fig. 8. This figure shows the intersection entry interfering lane gap.

If the traffic control for the intersection is signalized, the vehicle does not traverse the intersection in the current microsimulation timestep. Signalized intersections maintain internal queued buffers in which vehicles are placed to traverse the intersection. Each intersection has one queued buffer for each incoming lane.

If the conditions of the signalized traffic controller have been satisfied, a vehicle must check whether the appropriate buffer has space to receive the vehicle. (The intersection buffer's capacity is set by the configuration file key `CA_INTERSECTION_CAPACITY`.) If this is the case, the vehicle is removed from the incoming link and is placed in the intersection buffer for a wait period (specified by the configuration file key `CA_INTERSECTION_WAIT_TIME`).

After the time period has expired, the vehicle exits from the buffer to the first cell (or L^{th} cell if the vehicle has length L) on the destination link if the cell is vacant. If it is not, the vehicle waits in the intersection buffer until the cell becomes vacant.

The buffers have a fixed size, so that if the buffer is full the vehicle cannot enter the intersection and must wait on the link.

At unsignalized intersections, vehicles can enter and exit the intersection in a single timestep. Therefore, if the conditions of the unsignalized traffic controller have been satisfied for intersection entry, a vacant cell on the destination link in the destination lane must be available for the vehicle to enter the intersection. The vehicle's current speed is used to determine which cell to reserve on the destination link.

If the primary destination cell is unavailable, the next cell closer to the intersection is tried. This process continues until an available cell is found or until all the cells between the intersection and the primary destination cell are tried. A marker is placed in the destination cell to reserve the cell.

If a vehicle successfully reserves a place in the queue or on the next link, an internal state variable will be set to indicate that it can proceed. This variable is used during the movement procedure to determine whether to remove a vehicle from a link or decrease its speed. Vehicles traversing unsignalized intersections are placed on their destination link during the cleanup procedure at the end of a timestep.

3.7.1 Off-plan Vehicles

An off-plan vehicle is one that is not in an acceptable approach lane when it is ready to enter an intersection and thus cannot follow its assigned plan. Vehicles that have not moved for the number of timesteps defined by the configuration file key `CA_MAX_WAITING_SECONDS` also become off-plan.

The timestep when the vehicle tries to exit from the simulation is calculated using the off-plan exit time (configuration file key `CA_OFF_PLAN_EXIT_TIME`). Once this is calculated, a new destination link is selected from links connected to the vehicle's current lane.

New destination links are randomly selected for off-plan vehicles until the current timestep is equal to the calculated exit timestep. Once time is reached, the vehicles are removed from the simulation at the nearest parking place.

3.7.2 Abandon Plan

Vehicles attempting to enter an intersection (and that have not moved for a specified period of time) abandon their plans and, if possible, select a different destination link. The time period is defined by `CA_MAX_WAITING_SECONDS`.

These vehicles are marked as off-plan and are removed at the nearest parking place. Allowing vehicles to become off-plan after a specified waiting period is necessary to prevent traffic gridlock.

3.7.3 Movement

The movement rule is as follows: *accelerate when you can; slow down if you must; sometimes slow down for no reason*. The rule is executed to update the speed and position of each vehicle on the roadway.

A gap is defined as the distance between a vehicle and the next car ahead. Each vehicle tries to accelerate up to a desired speed if the gap is greater than the current speed. The desired speed is limited to the speed limit posted on each link and the maximum speed for each vehicle type and subtype (as specified in the vehicle prototype file).

If the gap is smaller than the current speed, the vehicle will slow down until its current speed is equal to the gap, thus imposing the no-collision condition. Each vehicle also has a random probability of slowing down. This is called the deceleration probability (P_D) (configuration file key `CA_DECELERATION_PROBABILITY`). Use of the deceleration probability is essential to produce realistic traffic dynamics (such as jam waves) from individual vehicle interactions.

To compute a vehicle's speed ($Vt+1$) and the next position on a link, first compute the speed based on the gap and the vehicle's speed in the current timestep (V_t) as follows:

- Compute Gap
- if $(V_t < Gap\ AND\ V_t < V_{Max})$
  $V_{t+1} = V + A_t$

The acceleration A_t is determined separately for each vehicle subtype. For autos, A_t is the maximum acceleration as specified in the vehicle prototype file. For other vehicles, acceleration is grade and velocity dependent.

Under the assumption of constant power acceleration, A_{Max} is interpreted as the maximum acceleration at V = 7.5 m/sec = 1 cell/timestep. Then, the velocity dependence is $A = A_{Max}/V$.

The grade dependence is handled by taking into account the acceleration caused by gravity, $A = A_{Max}/V - gsinq$, where q is the grade. Negative accelerations are possible, until a vehicle reaches its "crawl speed" of 1 cell/timestep. Fractional accelerations are handled by using the greatest integer part and adding 1 randomly. That is, an acceleration of 1.6 cells/timestep/timestep is implemented as an acceleration of 2 (60% of the time) and 1 (40% of the time).

Each moving automobile (*Speed > 0*), but not heavy vehicles, has a random probability of decelerating in each timestep. Compute the probability and slow down if the computed probability is less than the deceleration probability.

- if $(V_{t+1} > 0)$ and $(N_{Rand} < P_D)$
  $V_{t+1} = V_t + 1$

And finally, move the vehicle to its new grid position based on the new speed.

- $New\ Cell = Current\ Cell + V_{t+1}$

3.7.4 Entering Parking Places

To remove vehicles from the roadway at destination parking places, the Traffic Microsimulator checks all of the cells in all lanes downstream from the parking place for a distance of $V_{GlobalMax}$ cells.

If a vehicle is found on the last step of the current leg of its plan and with this parking place as its destination, the vehicle is removed from the roadway. Its ID is placed onto the list of vehicles present at that parking place.

3.8 Preparing for a Timestep

3.8.1 Update Signals

Timing tables provided for each signal are used to update them at each timestep. Signalized traffic controls are initialized at the beginning of the simulation to the first interval of the signal cycle's first phase when the signal offset is 0.0. When the offset is not zero, the signal is initialized to the phase and interval that corresponds to simulation time 0 in the offset cycle.

3.8.2 Prepare Nodes

Find vehicles in each intersection that are ready to be ejected during this timestep. Vehicles exit from the intersection queued buffers when their residence time in the buffer is greater than the intersection residence time specified by the configuration file key `CA_INTERSECTION_WAIT_TIME`.

Vehicles exit from the queued buffer onto the first cell in the destination lane on the destination link. Exiting vehicles reserve their destination cell before vehicles on links calculate movement, thus giving the vehicles exiting from intersection buffers precedence over vehicles on the links. This procedure places a temporary vehicle marker on the next grid for each vehicle that will leave the intersection on this timestep.

3.9 Cleaning Up After a Timestep

3.9.1 Migrate Vehicles

Any vehicle that has passed from a region of a link controlled by a CPU into a region controlled by its neighbor must be encoded in a message and sent to that neighbor. The Migrate Vehicles process is done on a link-by-link basis.

3.9.2 Migrate Travelers

Some travelers not in vehicles may have been placed in the Migrating Travelers list during the timestep. The Migrate Travelers procedure encodes those travelers into messages and passes them on to the desired CPUs, thus clearing out the list as it goes.

3.9.3 Clean up Nodes

The Clean up Nodes procedure causes each intersection to eject the first vehicle in each of its buffers into previously reserved locations on the destination link.

Vehicles are transferred from the buffers to their reserved destination cells during the cleanup phase, which takes place after movement changes for all the vehicles are executed. Vehicle speed does not change during intersection entry/exit at a signalized intersection. Vehicles are placed in the first cell on the destination link with the same velocity that they entered the intersection buffer.

3.9.4 Clean up Edges

The Clean up Edges procedure clears all temporary vehicle markers from the grids. In addition, if the cleanup action state variable for a vehicle is `eject`, it places the vehicle in the intersection buffer (if buffered; otherwise, place it directly onto the next edge).

If the cleanup action is `migrate`, it deletes the vehicle (which has already been sent to its destination CPU in the migration step).

3.10 Supporting Parallel Computation

The Traffic Microsimulator runs on multiple CPUs to maximize computational speed. Updating vehicle positions then can be done in parallel on individual CPUs. This method is faster than a single, sequential update algorithm on transportation networks with a large number of vehicles.

3.11 Transportation Network Partition

The transportation network is partitioned among the CPUs, with each CPU receiving a set of nodes and links (Fig. 9). To partition the network among the CPUs, an orthogonal bisection (OB) algorithm or the METIS graph-partitioning library can be used.

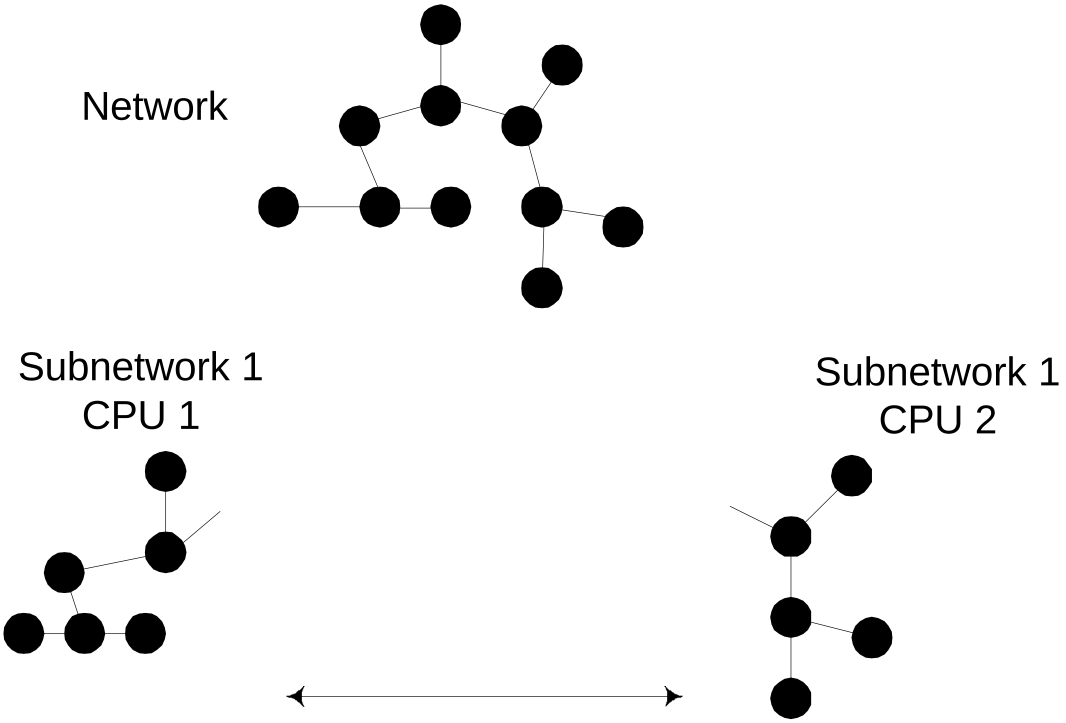

Fig. 9. A graphic representation of a transportation network partition.

METIS is a public domain package. Which algorithm is used is determined at run time by a combination of the configuration file keys `PAR_PARTITION_FILE`, `PAR_USE_METIS_PARTITION`, and `PAR_USE_OB_PARTITION`.

Both algorithms use a cost function for each node. METIS also uses a cost function for each link. These costs can be based on the number of cells on the links attached to the node if no other information is available. As the simulation runs, it collects information on the amount of CPU time devoted to processing each link and node. This information can be saved in a *Run Time Measurements* file, which can be used to assign costs to the links and nodes in subsequent partitioning calculations. The configuration file keys that control this process are `PAR_RTM_INPUT_FILE` and `PAR_RTM_PENALTY_FACTOR`. The Run Time Measurements file is placed in the directory named by `OUT_DIRECTORY` in a file named `RTM_FEEDBACK_FILE`.

Partitioning can be saved for later use by *DistributePlans* or a subsequent simulation run. This is controlled by the configuration file keys `PAR_SAVE_PARTITION` and `PAR_PARTITION_FILE`. If neither METIS nor OB partitioning has been requested, the simulation will look for this partition file.

3.12 Distributed Links and Boundary Information Flow

Links that connect nodes residing on different CPUs are split in the middle (Fig. 10). These links are distributed links. Each CPU is responsible for one-half of the link. Each distributed link is assigned a number of active grid cells belonging to a given CPU. Such an assignment is necessary to consistently divide links with an odd number of cells.

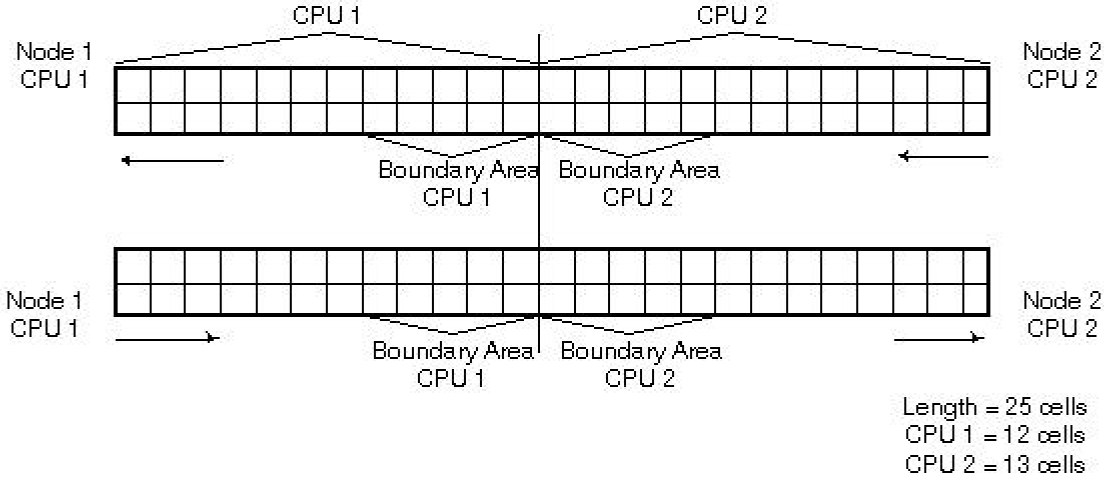

Fig. 10. This figure shows a distributed (split) link.

The area in the middle of the distributed links is called a boundary area. The boundary area's width is currently $V_{GlobalMax}$ (5) cells. Links shorter than `PAR_MIN_CELLS_TO_SPLIT` cells will not be split. The maximum distance (forward or backward on a link) that can be used for gap calculations is limited to the boundary width on distributed links.

Vehicles are transferred between CPUs as they traverse these split links. Each split link introduces a message-passing delay during the update sequence because messages must be passed between the CPUs for vehicles that are crossing split links. Two types of messages must be exchanged between CPUs with distributed links:

- Vehicle Migration Messages, which are messages for vehicles transferred to the other part of the link on a different CPU.
- Boundary Exchange Messages, which are messages containing information about vehicle positions in the boundary area of a link.

Vehicle migration messages occur for all vehicles that have traversed half the cells of a distributed link. These vehicles must be transferred to the CPU that owns the other half of the distributed link. All information about the vehicle, its occupants, and their plans is put into a message and sent to the destination CPU, after which the vehicle is removed from the originating CPU.

Upon receipt of the message, the destination CPU recreates the vehicle and travelers using the information in the message; it then places them at the appropriate position on its half of the distributed link.

Exchange of boundary information between CPUs is called a boundary exchange. Boundary exchange messages are necessary to correctly calculate position changes (movement and lane changes) for vehicles in a CPU's boundary area. Information about vehicles in the next $V_{GlobalMax}$ cells (or preceding $V_{GlobalMax}$ cells, depending on the direction of traffic flow) is necessary to execute the appropriate gap calculations for lane changes and movement.

Each CPU maintains a list of its distributed links and of the CPU owners of the other half of the links. Boundary exchanges must be conducted before lane changes and again before vehicle movement. Each CPU initiates the exchanges at the appropriate time. Each CPU waits until it receives all of the boundary exchange messages from neighboring CPUs.

Comparison of Fig. 11with Fig. 3 shows how the message passing is interleaved with the simulation update processing.

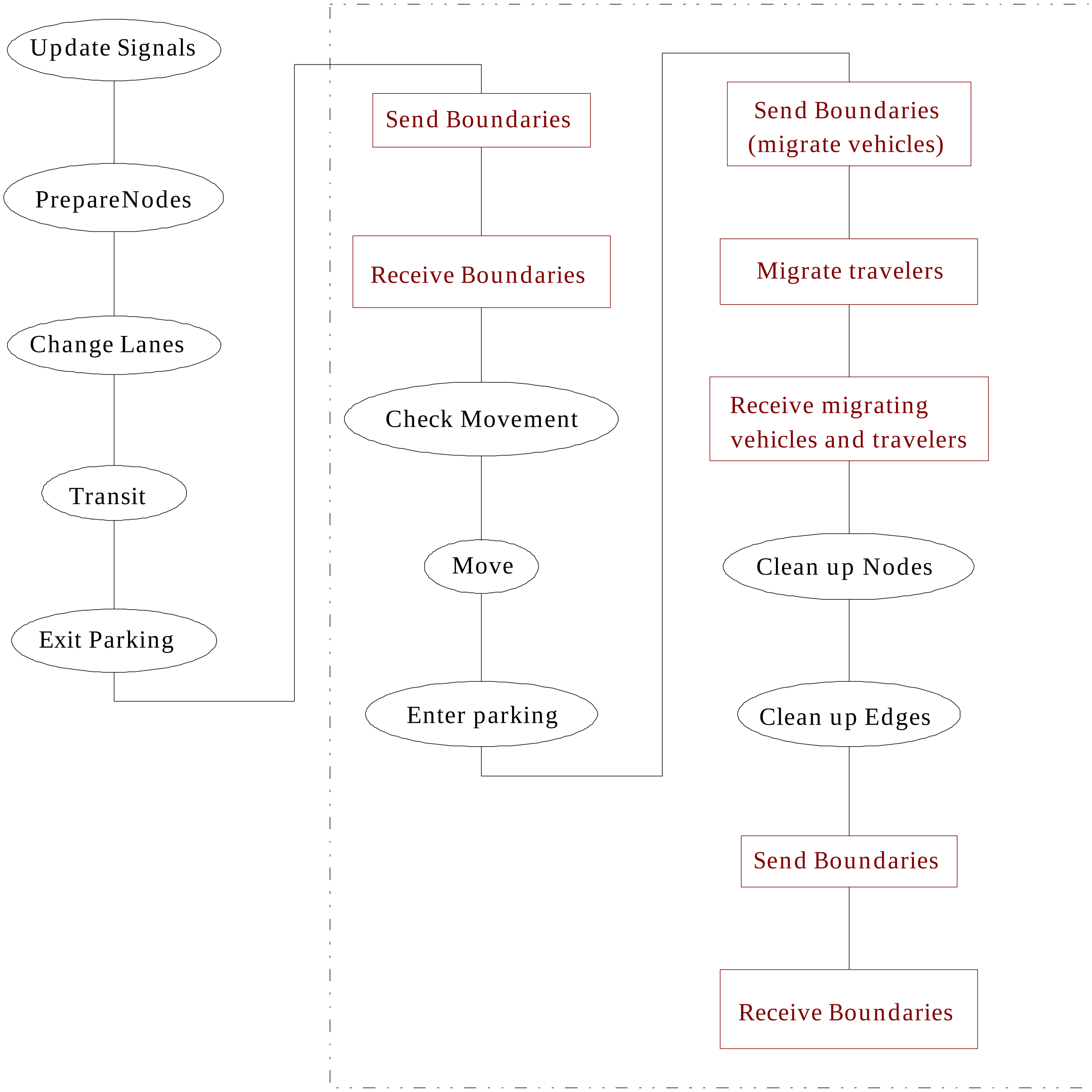

Fig. 11. The rectangular (red) boxes show information flow in a distributed version of the Traffic Microsimulator.

3.12.1 Parallel Computation Sequence and Synchronization Points

The Traffic Microsimulator, a distributed object simulation, uses a master/slave paradigm. The master process starts the slave processes, handles the initialization sequence, and serves as a synchronization point for the slave processes.

The slave processes do all of the work in the simulation. After initialization, each slave process completes successive update cycles until the end of the specified simulation run. The slave processes synchronize with the master process at the beginning of each timestep or at the beginning of a sequence of timesteps, depending on the value of the configuration file key `CA_SEQUENCE_LENGTH` (see Fig. 11). The message traffic among the master and slaves for controlling all of these tasks is shown in Fig. 12and Fig. 13.

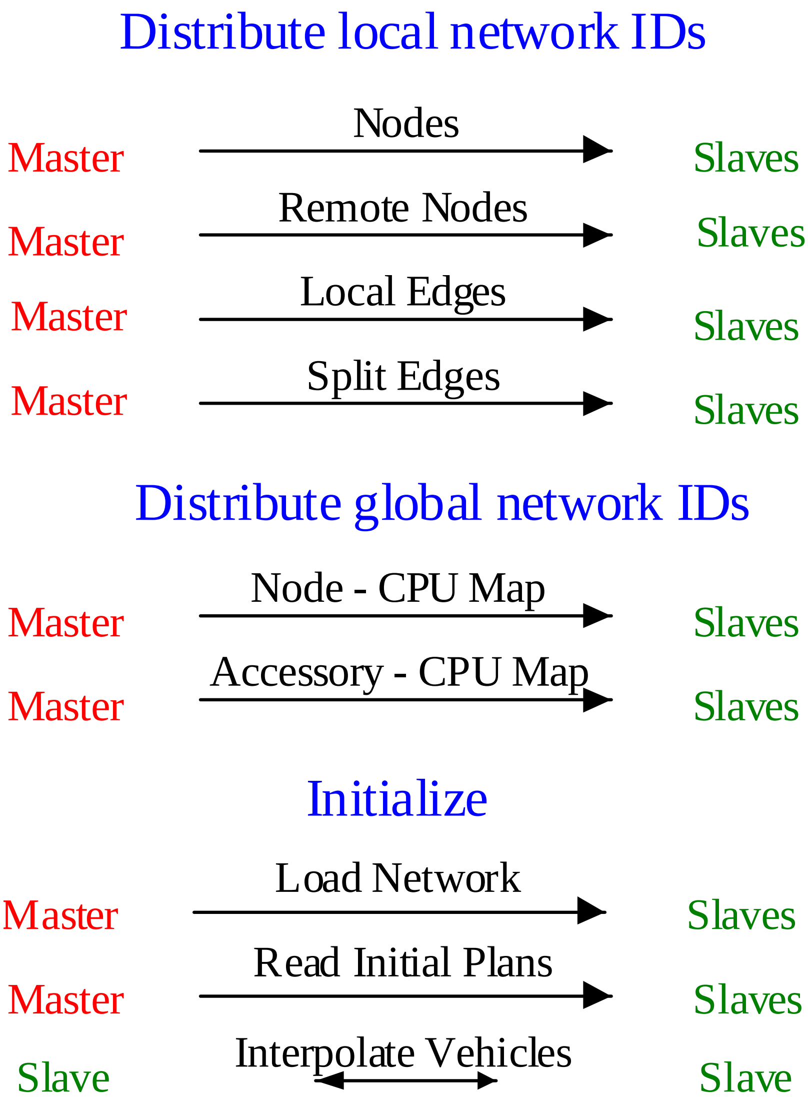

Fig. 12. Initialization message traffic.

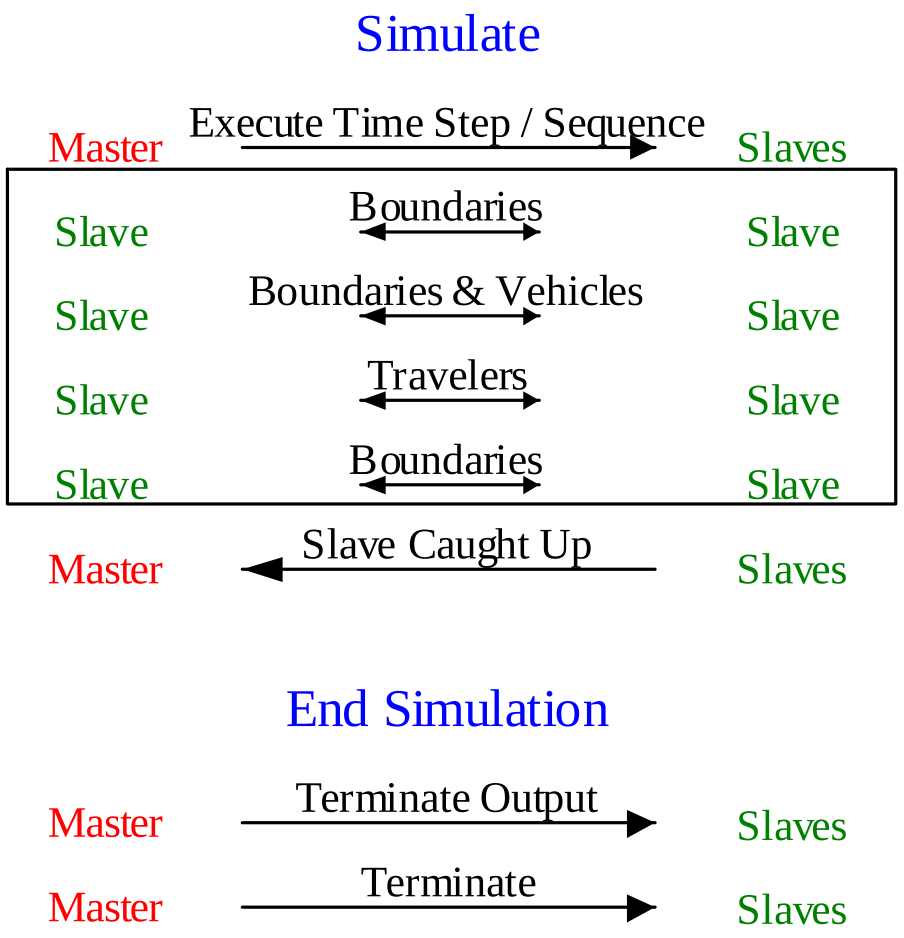

Fig. 13. The top part of this figure shows the simulation message traffic. The set of messages in the box is repeated once for every timestep up to the number of steps in the sequence. The lower part of this figure shows termination message traffic.

3.12.2 Initialization Sequence

The master process begins by reading the network information from the database, constructing a copy of the transportation network, and constructing or reading a partition. The master then is ready to create and initialize the following five-step slave process:

Step One Start slave processes.

Step Two Send each slave ID lists of its local nodes and links and lists of those connected to it by distributed links.

Step Three Send each slave a mapping from node IDs to CPU IDs, and optionally (depending on the setting of the configuration file key `CA_BROADCAST_ACC_CPN_MAP`) a mapping from accessory IDs to CPU IDs.

Step Four Tell each slave to construct its transportation subnetwork from database information.

Step Five Tell slaves to read in the initial plans, queue initial vehicles on parking places, and initially place vehicles on the links at the given simulation start time.

3.12.3 Simulation

After the initialization sequence is complete, the master starts the simulation by telling the slaves to execute the first timestep. The master process waits until all of the slaves complete execution of a fixed number of timesteps. It then sends a message to the slaves to execute the next timestep sequence.

3.12.4 Termination

The master sends messages to the slaves that tell them to shut down the parallel I/O system and then to exit when the requested number of timesteps has been executed.

3.12.5 Overlapping Computation and Communication

For efficiency, a parallel code should overlap communication whenever possible. This enables a CPU to continue executing useful work while waiting for responses from other CPUs.

The Traffic Microsimulator accomplishes this by noting which links are under a single CPU’s control and which are shared. After sending boundary information, each CPU can update all of its non-shared links before it must make use of boundary information received from other CPUs. Fig. 14 shows the sequences of computation and communication in such a modified timestep execution. If the configuration file key `CA_LATE_BOUNDARY_RECEPTION` is set, the simulation will arrange computations in this manner.

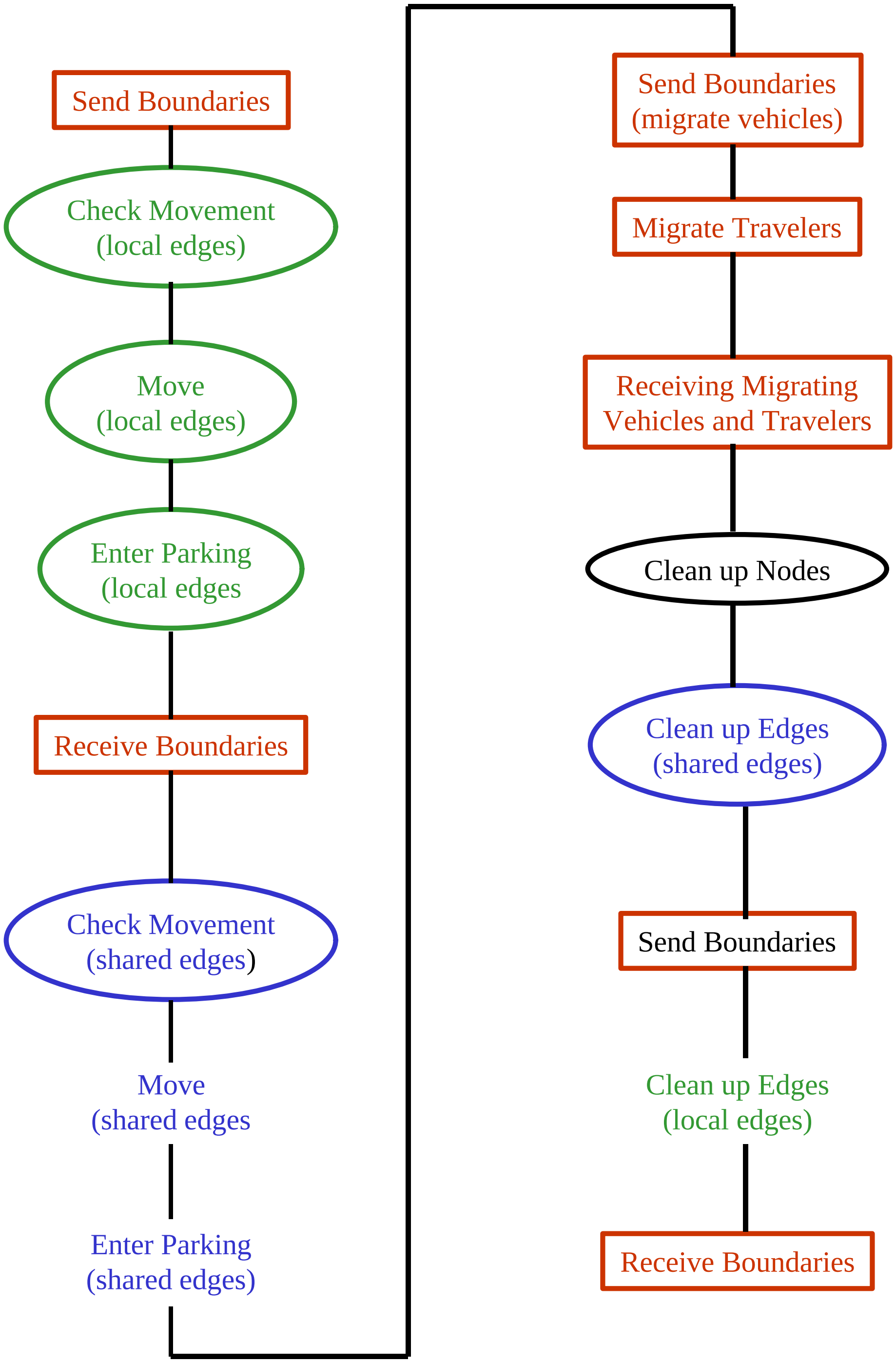

Fig. 14. Modifications to the processes enclosed in the dashed box support overlapping computation. Communication computation on local edges (in green) takes place while a CPU waits for information about shared edges (in blue).

3.12.6 Output Collection

Slaves generate in parallel all output information from the Traffic Microsimulator. Each slave sends a message to the master indicating what sort of information it would like to write and how many bytes the information will require on disk.

The master collates the requests from all the slaves and responds to each, indicating an offset into a file for writing the information. Each slave then writes its information to disk at the indicated location. The message traffic generated by the output system is not shown in Fig. 12 or Fig. 13clarity.

4. Simulation Output Files

4.1 Overview

This TRANSIMS Simulation Output subsystem collects data from a running microsimulation and stores it for subsequent examination by the analyst or use by other TRANSIMS software components. It provides a software layer that insulates applications from the details of the file structure and provides great flexibility in the specification of the data to be collected.

A parallel communication library is used to collect data in ASCII format into a single file written by the master simulation process. No postprocessing is required with this mechanism.

4.2 File Format

This section describes the file formats of the eight types of simulation outputs currently implemented. All fields are described, but the filtering capability enables suppression of any output field for which the analyst has no interest, thus resulting in smaller output files.

Applications that read the output produced by the simulation should always use the functions for reading. The functions provided by the output representation automatically handle records with suppressed fields and only attempt to read the fields that were actually written. This enables the implementation of general postprocessing applications that need not be cognizant of the number and order of the fields written by the simulation.

4.3 Traveler Event

The Traffic Microsimulator outputs traveler event records each time an event of interest to the analyst takes place. The simulation time interval during which to record events is defined in the input configuration file.

Filtering capabilities are provided so that the analyst can select which of the many potentially interesting events should be recorded. Table 2 provides a list of events that may be of interest; these are specified in the STATUS and ANOMALY output fields. The other fields describe the traveler's state at the time the event took place.

Table 2. Traveler event record fields.

Field	Description
TIME	The current time (seconds from midnight).
TRAVELER	The traveler ID.
TRIP	The traveler's trip ID.
LEG	The traveler's plan leg ID.
VEHICLE	The vehicle ID; value = 0 if not in a vehicle.

Field	Description
VEHTYPE	The vehicle type: 0 = walk 1 = auto 2 = truck 3 = bicycle 4 = taxi 5 = bus 6 = trolley 7 = streetcar 8 = light rail 9 = rapid rail 10 = regional rail
VSUBTYPE	The vehicle subtype may be unused; value = 0 if not applicable.
ROUTE	The transit route ID; value = -1 if not in a transit vehicle.
STOPS	The count of number of stop signs encountered on current plan leg.
YIELDS	The count of number of yield signs encountered on current plan leg.
SIGNALS	The number of traffic signals encountered on current plan leg.
TURN	The type of last turn made: 0 = straight direction (no turn) 1 = right turn -1 = left turn 2 = hard right turn -2 = hard left turn values 3 to 6 represent increasingly more extreme right turns values –3 to –6 represent increasingly more extreme left turns -7 = reverse direction (U-turn)
STOPPED	The time (seconds) spent stopped on current plan leg.
ACCELS	The time (seconds) spent accelerating from 0 on current plan leg.
TIMESUM	The total time (seconds) spent on current plan leg.
DISTANCESUM	The total distance (meters) traveled on current plan leg (see accompanying text for more information).
USER	The analyst-defined field: any integer value is acceptable; definition may vary with each case study.
LINK	The link ID when traveler is on a link or previous link when traveler is in an intersection.
NODE	The node ID traveler is traveling away from on a link or node traveler is in an intersection.

Field	Description
ANOMALY	The type of anomaly: 0 = no anomaly occurred 1 = traveler is off plan 2 = traveler cannot find next link in plan 3 = traveler cannot find next parking place in plan 4 = traveler cannot find next vehicle in plan 5 = traveler cannot find next transit stop in plan 6 = traveler cannot board full transit vehicle 7 = driver of transit vehicle skipped stop that had passengers waiting to board 8 = driver of vehicle cannot change lanes because of congestion

Field	Description
STATUS	The traveler's current status bits: (see accompanying text for a detailed explanation of status bit interpretation). 0x1 = traveler is on a link (persistent) 0x2 = change in traveler's on-link status 0x4 = traveler is on a leg (persistent) 0x8 = change in traveler's on-leg status 0x10 = change in traveler's on-trip status 0x20 = traveler is non-motorized, i.e., walking, bicycling (persistent) 0x40 = traveler is not in the study area (persistent) 0x80 = change in traveler's in-study area status 0x100 = traveler is in a vehicle (persistent) 0x200 = change in traveler's vehicle occupancy status 0x400 = traveler is the driver (persistent) 0x800 = change in traveler's driver status 0x1000 = traveler is waiting at some location (persistent) 0x2000 = change in traveler's waiting status 0x4000 = location is a parking place (persistent) 0x8000 = location is a transit stop (persistent) 0x10000 = driver of transit vehicle is at a transit stop (persistent) 0x20000 = change in driver's transit vehicle at stop status 0x40000 = driver of transit vehicle is on a layover (persistent) 0x80000 = change in driver's transit vehicle on layover status 0x100000 = driver's transit vehicle is full (persistent) 0x200000 = change in driver's transit vehicle full status 0x400000 = traveler is off plan (persistent) 0x800000 = change in traveler's off-plan status 0x1000000 = beginning of simulation 0x2000000 = end of simulation 0x4000000 = location is an activity location (persistent) 0x8000000 = undefined 0x10000000 = undefined 0x20000000 = undefined 0x40000000 = undefined 0x80000000 = undefined

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>LOCATION</td><td>The traveler’s location: parking place ID, transit stop ID, or activity location ID, depending on the event as defined as follows:
<table>
<tr><th>EVENT</th><th>LOCATION value</th></tr>
<tr><td>Begin/End plan leg</td><td>parking place ID or transit stop ID</td></tr>
<tr><td>Begin/End trip</td><td>parking place ID or transit stop ID</td></tr>
<tr><td>Enter/Exit vehicle</td><td>parking place ID or transit stop ID</td></tr>
<tr><td>Begin/End driving</td><td>parking place ID or transit stop ID</td></tr>
<tr><td>Waiting for transit</td><td>transit stop ID</td></tr>
<tr><td>Waiting at parking</td><td>parking place ID</td></tr>
<tr><td>Begin/End activity</td><td>activity location ID</td></tr>
<tr><td>Transit vehicle at stop</td><td>transit stop ID</td></tr>
<tr><td>Transit vehicle on layover</td><td>transit stop ID</td></tr>
<tr><td>Transit vehicle full</td><td>transit stop ID</td></tr>
<tr><td>Can’t find parking</td><td>parking place ID</td></tr>
<tr><td>Can’t find vehicle</td><td>parking place ID</td></tr>
<tr><td>Can’t find transit stop</td><td>transit stop ID</td></tr>
<tr><td>Can’t board transit</td><td>transit stop ID</td></tr>
<tr><td>Skipped transit stop</td><td>transit stop ID</td></tr>
</table>
When the traveler is on a link or in an intersection, the LOCATION field is zero.</td></tr>
</table>

The `STATUS` field is bit-oriented. Each bit represents a characteristic about the traveler that is true whenever the bit is set. Multiple bits set means that multiple characteristics are true at this time. Interpretation of the `STATUS` field involves determining which combination of characteristics is currently true according to the table that describes the individual bits. It is convenient to view the `STATUS` field in hexadecimal notation because this more clearly illuminates the patterns in the field.

Status values are generally represented in bit pairs. The lower bit of a pair is termed the “persistent bit,” whereas the upper bit is termed the “change bit.”

- The persistent bit is set during the entire time that the condition is true.
- The change bit is set only for the timestep when a change in the persistent bit occurs.

This scheme enables the analyst to identify the beginning and the end of a persistent condition without comparing multiple events.

For example, when a traveler begins a leg, the persistent bit representing `on leg` (0x4) is set, and the change bit representing `change in on leg` (0x8) is set. While the traveler is on the leg, the persistent bit (0x4) remains set, and the change bit (0x8) is cleared. When the traveler ends the leg, the persistent bit (0x4) is cleared, and the change bit (0x8) is again set for one timestep. While the traveler is not on a leg (e.g., while waiting somewhere), both the persistent bit and the change bit are cleared.

A few of the status bits take place singly rather than in pairs because both bits are not required. For example, a persistent bit for `on trip` is not needed because travelers are only simulated while they are on a trip. A persistent bit that is always set provides no additional information and clutters the output, and therefore it is not used. The `non-motorized` bit (0x20) is used in conjunction with the `on leg` bits to indicate that the leg does not involve vehicular travel. The `location type identification` bits (0x4000, 0x8000, and 0x4000000) are used in two ways:

1) They are used in conjunction with bits 0x1000 and 0x2000 to identify the type of location at which the traveler is waiting.

2) They are also used to specify the type of location when the `LOCATION` field represents a parking place or transit stop ID. For example, when a traveler begins a leg at a parking place, bit 0x4000 will be set in addition to bits 0x4 and 0x8 to signify that the beginning location of the leg is a parking place.

The `DISTANCESUM` field accumulates the distance traveled along links and within intersections. Upon entering the intersection, `DISTANCESUM` is incremented by the setback on the link just left; and when exiting the intersection, `DISTANCESUM` is incremented by the setback on new link.

4.4 Snapshot Data

Snapshot data provides detailed information about the state of the simulation at a point in time. Multiple snapshots allow following the evolution of the simulation state through time. Snapshot data may be viewed with the Output Visualizer.

4.4.1 Vehicle Snapshot Data

Vehicle snapshot data provide information about vehicles traveling on a link. When collected for every link on every timestep, such data give a complete trajectory for each vehicle in the simulation. Vehicle snapshot data are collected as frequently as the analyst indicates in the input configuration file for the specified links. Table 3 provides a list of vehicle snapshot record fields.

Table 3. Vehicle snapshot data record fields.

Field	Interpretation
`VEHICLE`	The vehicle ID.
`TIME`	The current time (seconds from midnight).
`LINK`	The link ID on which the vehicle was traveling.
`NODE`	The node ID from which the vehicle was traveling away from.
`LANE`	The number of the lane on which the vehicle is traveling.
`DISTANCE`	The distance (in meters) the vehicle is away from the setback of the node from which it is traveling away.
`VELOCITY`	The velocity (in meters per second) of the vehicle.

Field	Interpretation	
VEHTYPE	The vehicle type: 0 = walk 1 = auto 2 = truck 3 = bicycle 4 = taxi 5 = bus	 6 = trolley 7 = streetcar 8 = light rail 9 = rapid rail 10 = regional rail
ACCELER	The acceleration (in meters per second) the vehicle had in the current timestep.	
DRIVER	The driver ID.	
PASSENGERS	The count of passengers in vehicle.	
EASTING	The vehicle's x-coordinate (in meters).	
NORTHING	The vehicle's y-coordinate (in meters).	
ELEVATION	The vehicle's z-coordinate (in meters).	
AZIMUTH	The vehicle's orientation angle (degrees from east in the counterclockwise direction).	
USER	The user-defined field that can be set on a per-vehicle basis.	

4.4.2 Intersection Snapshot Data

Intersection snapshot data provide information about a vehicle as it traverses an intersection. These data are collected as frequently as the analyst indicates in the input configuration file for the specified nodes. Table 4 provides a list of intersection snapshot record fields.

Table 4. Intersection snapshot data record fields.

Field	Interpretation
VEHICLE	The vehicle ID.
TIME	The current time (seconds from the midnight).
NODE	The node ID where the vehicle is located.
LINK	The link ID from which the vehicle entered.
LANE	The number of the lane from which the vehicle entered.
QINDEX	The vehicle position in the intersection buffer.

4.4.3 Traffic Control Snapshot Data

Traffic control snapshot data report the current state of the traffic signal at a node. These data are collected as frequently as the analyst indicates in the input configuration file for the specified nodes. Table 5 lists traffic control snapshot record fields.

Table 5. Traffic control snapshot data record fields.

Field	Interpretation
NODE	The node ID, where the signal is located.
TIME	The current time (seconds from midnight).
LINK	The link ID entering the signal.
LANE	Number of the lane entering the signal.
SIGNAL	The type of control present: 0 = None 1 = Stop 2 = Yield 3 = Wait 4 = Caution 5 = Permitted 6 = Protected 7 = Permitted after stop

4.5 Summary Data

Summary data reports aggregate data about the simulation. Summary data is sampled, accumulated, and reported periodically throughout the simulation.

The first record in each summary data file contains metadata information about the parameters used in data collection. The metadata record begins with the keyword METADATA, followed by the date on which the file was created. Other items in the metadata record follow in the form of keyword-value pairs. The metadata items unique to each summary data file are described in the following sections.

4.5.1 Link Travel Times Summary Data

Link travel time summary data report vehicle counts and travel times on links accumulated as vehicles exit the links. These data are collected as frequently as the analyst indicates in the input configuration file for the specified links.

The metadata for the link travel time summary file is the TIME_STEP configuration file key (i.e., the frequency at which data is reported).

There are separate data records for each turning movement leaving each lane on the link. Table 6 lists link travel times summary field records.

Table 6. Link travel times summary data field records.

Field	Interpretation
LINK	The link ID being reported.
NODE	The node ID from which the vehicles were traveling away.
TIME	The current time (seconds from midnight).
COUNT	The number of vehicles leaving the link.
SUM	The sum of the vehicle travel times (in seconds) for vehicles leaving the link. (The time spent in the previous intersection is included in this value.)
SUMSQUARES	The sum of the vehicle travel time squares (in seconds squared) for vehicles leaving the link. (The time spent in the previous intersection is included in this value.)
TURN	The type of turn the vehicle made when leaving the link: 0 = straight direction (no turn) 1 = right turn -1 = left turn 2 = hard right turn -2 = hard left turn values 3 to 6 represent increasingly more extreme right turns values –3 to –6 represent increasingly more extreme left turns -7 = reverse direction (U-turn)
LANE	The lane number.
VCOUNT	The number of vehicles on the link.
VSUM	The sum of vehicle velocities (in meters per second) on the link.
VSUMSQUARES	The sum of the squares of the vehicle velocities (in meters squared per second squared).

4.5.2 Link Densities Summary Data

Link density summary data report vehicle counts and velocities within “boxes” that partition the link. These data are collected as frequently as the analyst indicates in the input configuration file for the specified links.

The metadata for the link density summary file are the TIME_STEP, SAMPLE_TIME, and BOX_LENGTH configuration file keys.

There are separate data records for each lane on the link. The box length is specified in the input configuration file. Table 7 lists link densities summary record fields.

Table 7. Link densities summary data record fields.

Field	Interpretation
LINK	The link ID being reported.
NODE	The node ID from which the vehicles were traveling away.
DISTANCE	The ending distance of the box (in meters) from the setback of the node from which the vehicles were traveling away.
TIME	The current time (seconds from midnight).
COUNT	The number of vehicles in the box.
SUM	The sum of the vehicle velocities (in meters per second) in the box.
SUMSQUARES	The sum of the squares of the vehicle velocities (in meters squared per second squared).
LANE	The lane number.

4.5.3 Link Velocities Summary Data

Link velocity summary data report histograms of vehicle velocities within “boxes” that partition the link. These data are collected as frequently as the analyst indicates in the input configuration file for the specified links.

The metadata for the link velocity summary file are the TIME_STEP, SAMPLE_TIME, BOX_LENGTH, VEHICLE_TYPE, VEHICLE_SUBTYPE, VELOCITY_MAX, VELOCITY_BINS, and CELL_LENGTH configuration file keys.

The input configuration file specifies the box length, number of histogram bins, and maximum velocity. The maximum velocity is typically 37.5 m/s and the velocity range is divided into five bins, in addition to an overflow bin that extends to infinity. Histogram intervals are defined to be closed at the lower end of the bin and open at the upper end. Table 8 lists link velocities summary record fields.

Table 8. Link velocities summary data record fields.

Field	Interpretation
LINK	The link ID being reported.
NODE	The node ID from which the vehicles were traveling away.
DISTANCE	The ending distance of the box (in meters) from the setback of the node from which the vehicles were traveling away.
TIME	The current time (seconds from midnight).
COUNT0	The number of vehicles with velocities in the range [0, 7.5).
COUNT1	The number of vehicles with velocities in the range [7.5, 15).
COUNT2	The number of vehicles with velocities in the range [15, 22.5).
COUNT3	The number of vehicles with velocities in the range [22.5, 30).
COUNT4	The number of vehicles with velocities in the range [30, 37.5).
COUNT5	The number of vehicles with velocities in the range [37.5, infinity).

4.5.4 Link Energy Summary Data

Link energy summary data report histograms of vehicle energies (integrated power) accumulated as vehicles enter the links. Energy is defined as the sum of the vehicle's power over each timestep, where power is defined as the velocity times the acceleration when the acceleration is greater than zero.

The metadata for the link energy summary file are the TIME_STEP, ENERGY_SOAK, ENERGY_MAX, ENERGY_BINS, SHORT_SOAK_TIME, MEDIUM_SOAK_TIME, and LONG_SOAK_TIME configuration file keys.

Vehicles are assumed to have zero power while they are at intersections. The units for energy are cells-squared per second-squared.

These data are collected as frequently as the analyst indicates in the input configuration file for the specified links. The number of histogram bins and maximum energy is specified in the input configuration file. Histogram intervals are defined to be closed at the lower end of the bin and open at the upper end. Table 9 lists link energy summary record fields.

Table 9. Link energy summary data record fields.

Field	Interpretation
LINK	The link ID being reported.
NODE	The node ID from which the vehicles were traveling away.
TIME	The current time (seconds from midnight).
ENERGY0	The number of vehicles with integrated power in the range [0, *energy_maximum* / *number_bins*).
ENERGY1	The number of vehicles with integrated power in the second bin.
ENERGY2	The number of vehicles with integrated power in the third bin.
ENERGYn	The number of vehicles with integrated power in the range [energy_maximum, infinity).

4.6 Output Filtering

A variety of output filtering capabilities have been designed to limit potentially voluminous output to only those items of interest in a particular simulation run. An unlimited number of output specifications may be included in the simulation configuration file, allowing for very fine-grained control of the output produced.

Time-based filtering may be used to restrict data collection to a subset of the total run time by specifying starting and ending times. The analyst specifies in the input configuration file the frequency of reporting for evolution and summary data and the sampling frequency for summary data.

Collected data may be restricted to a subset of nodes and links in the road network. Table 10 describes the field in the node specification file, and Table 11 describes the field in the link specification file. Regional filtering allows the specification of the corners of a rectangular region in which data should be collected. (Note that the Traffic Microsimulator does not currently use regional filtering.)

Table 10. Node specification fields.

Field	Description
NODE	The node ID.

Table 11. Link specification fields.

Field	Description
LINK	The link ID.

Data may be filtered by value, with only those items that pass all filters appearing in the output. The supported operators for value filtering are indicated in Table 12. Data fields in a record may be suppressed, resulting in shorter records.

Table 12. Value filtering operators.

Operators	Interpretation
==	equal to
!=	not equal to
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
%	an integer multiple of
!%	not an integer multiple of
@	included in the list (a list is a string of values starting with the left-bracket character ([ ), ending with the right-bracket character ( ]), and where each value is separated by the pipe character (|))
!@	not included in the list
&	has set bits
!&	has cleared bits

4.7 Utility Programs

4.7.1 *InterpretStatus.awk* Utility

The *InterpretStatus.awk* utility displays the STATUS field in the traveler event output data as a bit pattern for easier interpretation.

Usage:

```
InterpretStatus.awk <event file>
```

InterpretStatus.awk reads the event file and writes the bit patterns representing the STATUS field to standard output. The output may be redirected to a file (if preferred).

4.7.2 *SetupOutput* Utility

The *SetupOutput* utility copies a set of empty and test output tables into a specified directory. It takes the name of the directory as its only argument.

4.7.3 *CleanupOutput* Utility

The *CleanupOutput* utility removes a set of tables created by *SetupOutput*. It takes the name of the directory as its argument.

4.8 Files

Table 13 lists simulation output library files.

Table 13. Simulation output library files.

Type	File Name	Description
Binary Files	*libTIO.a*	The TRANSIMS Interfaces library.
Source Files	*outio.c*	The simulation output data structures and interface functions.
	outio.h	The simulation output interface functions source file.
Utilities	*InterpretStatus.awk*	Interprets event status field.
	SetupOutput	Creates empty and test output files.
	CleanupOutput	Removes empty and test output files.

4.9 Configuration File Keys

In the simulation output keywords, the trailing *n* must be replaced by an integer, beginning with 1 for the first set of output of each type (snapshot, event, and summary). If more than one set of output is desired for a particular type, the second set of keywords ends with *n*=2; the third set uses *n*=3, etc. There is no restriction to the number of output data sets of each type that may be requested. Default values can be specified for most of the simulation output keywords. Defaults are particularly useful when multiple output files are collected and the values of some keywords are the same for all files. The keywords for specifying default values are described in Appendix E, which follows the descriptions of the output keywords. The user may override any specified default by providing a value for the full keyword in the individual output specifications. The use of default values is optional.

- Appendix A lists the configuration file keys that specify microsimulation parameters.
- Appendix B lists configuration file keys that pertain to the snapshot (evolution) type of output.
- Appendix C lists configuration file keys that pertain to the event type of output.
- Appendix D lists configuration file keys that pertain to the summary type of output.

- Appendix E lists configuration file keys used by the *CompareDensity* and *CompareVelocit*y programs. Only the first of these keys is used by *CompareVelocity*.
- The configuration file keys in Appendix E are used to provide default values for many of the keywords described in Appendixes B, C, and D.

Appendix A: Configuration File Keys

Configuration File Key	Description
CA_BROADCAST_ACC_CPN_MAP CA_BROADCAST_TRAVELERS	If Broadcast Travelers is set, migrating travelers are broadcast to every CPU. Because only one CPU will eventually make use of the traveler, this is inefficient. If Broadcast Acc CPN Map is set, each CPU knows which CPU is associated with every accessory, so traveler migration messages can be targeted to only the single CPU that needs them. If the CPN Map is not broadcast, travelers must be broadcast.
CA_DECELERATION_PROBABILITY	To enhance traffic variation, each automobile driver randomly decides whether to decelerate for no apparent reason at each timestep. The probability of decelerating is a value in the range 0.0 to 1.0. Default = 0.2
CA_ENTER_TRANSIT_DELAY CA_EXIT_TRANSIT_DELAY	These keys specify the mean number of timesteps it takes for a single traveler to enter or exit a transit vehicle.
CA_GAP_VELOCITY_FACTOR	At unsignalized intersections and during protected movements at signalized intersections, drivers wait for a suitable gap in cross traffic before proceeding through the intersection. The number of empty cells in a suitable gap is based on the speed of the cross traffic and the gap velocity factor. The suitable gap is calculated for each lane of the cross traffic. Gap = Speed of Oncoming Vehicle * Gap Velocity Factor The gap velocity factor must be greater than 0.0. The default value is 3.0. Note that vehicles with a speed of 0 result in a suitable gap size of 0, which improves traffic flow in congested conditions.
CA_IGNORE_GAP_PROBABILITY	Drivers at unsignalized intersections wait for a suitable gap in cross traffic before proceeding through the intersection. Allowing each driver to ignore the gap constraint with some probability prevents the deadlock that would take place when vehicles are waiting for each other at multiway stop/yield signs. The probability that the drivers at multiway stop/yield signs will ignore the constraint is a value in the range of 0.0 to 1.0. Default = 0.66
CA_INTERSECTION_CAPACITY	Intersection Capacity determines the number of vehicles that can be held by each intersection’s buffers.

Configuration File Key	Description
CA_INTERSECTION_WAIT_TIME	`Intersection Wait Time` specifies the number of seconds that a vehicle requires to pass through a signalized intersection. A vehicle resides in an intersection-queued buffer for this amount of time and is then placed on the next link if the first cell on that link is unoccupied. It will remain in the intersection for a longer time if entry to the next link is blocked by another vehicle. Valid values are positive. Default = 1 second
CA_LANE_CHANGE_PROBABILITY	Variation in traffic is reduced by not allowing every driver who would change lanes based on vehicle speed and gaps in the traffic to do so at each timestep. This is done to prevent *lane hopping*. The probability that a driver will change lanes when speed and gaps permit is a value in the range of 0.0 to 1.0. Default = 0.99
CA_LATE_BOUNDARY_RECEPTION	If `Late Boundary Reception` is set, the simulation will try to overlap computation and communication.
CA_LONG_SOAK_TIME	The boundary (in seconds) between medium and long soak times for energy output. Default = 9000
CA_LOOK_AHEAD_CELLS	The preferred lane for a vehicle to be in as it approaches an intersection depends on the connectivity from the current link to the next link in the plan. In some situations, it is advantageous for the driver to look beyond the next link to subsequent links in the plan when deciding the preferred lane. Look Ahead Cells controls how far ahead the driver will look. A value of 0 indicates that the driver will not look beyond the next link. A positive value indicates that the driver will look at least one additional step beyond the next step in the plan. The number of additional links considered is determined by the lengths of the subsequent links, with link lengths being summed until the accumulated distance is greater than or equal to Look Ahead Cells. Valid values are positive or zero. Default = 35 cells
CA_MAX_WAITING_SECONDS	`Max Waiting Seconds` determines the number of seconds that a vehicle will try to enter an intersection. If the vehicle has not moved from the link into or through the intersection in `Max Waiting Seconds`, the vehicle abandons its plan and tries an alternative movement through the intersection (if one exists). `Max Waiting Seconds` must be > 0 and should be greater than the longest red phase of the traffic controls in the simulation. Default = 600 seconds
CA_MEDIUM_SOAK_TIME	The boundary (in seconds) between short and medium soak times for energy output. Default = 1800 seconds

Configuration File Key	Description
CA_NO_TRANSIT	If this flag is set, travelers whose plans originate or end at a transit stop are removed from the simulation. None of their remaining legs are used. (The transit driver plans do not fall into this category, thus transit vehicles can still be present in the simulation, but no passengers will use them.)
CA_OFF_PLAN_EXIT_TIME	`Off Plan Exit Time` specifies the number of seconds a vehicle is allowed to deviate from its plan before being removed from the simulation. This prevents off-plan vehicles from wandering on the transportation network. Valid values are positive. Default = 1 second
CA_PLAN_FOLLOWING_CELLS	`Plan Following Cells` specifies a count of the number of cells preceding the intersection within which a vehicle will make lane changes to get in an appropriate lane and thus transition to the next link in its plan. Beyond this distance, lane-changing decisions are based only on vehicle speed and gaps in the traffic. Within this distance, the lane required by the vehicle's plan is also taken into account. As the vehicle nears the intersection, the bias to be in the lane required to stay on plan is increased. Valid values are positive or zero. Default = 70 cells
CA_RANDOM_SEED	These three values are combined to initialize the random number generator. Note that the actual sequence of random numbers generated on a slave also depends on the number of slaves and the partitioning in general.
CA_SHORT_SOAK_TIME	The boundary (in seconds) between negligible and short soak times for energy output. Default = 600 seconds
CA_SEQUENCE_LENGTH	The slaves are implicitly synchronized among themselves by the actions of passing boundaries and migrating vehicles. They are also explicitly synchronized by the master every Sequence Length timestep. It may be more efficient to allow the implicit synchronization to control the simulation.
CA_SIM_START_HOUR CA_SIM_START_MINUTE CA_SIM_START_SECOND	These values are combined to calculate the simulation's starting time. Plans whose estimated arrival time is before the start time are not executed.
CA_SIM_STEPS	The simulation executes Sim Steps timesteps before exiting.
CA_SLAVE_MESSAGE_LEVEL CA_MASTER_MESSAGE_LEVEL	Only warning messages whose severity is at least as high as `Message Level` will be written to the master or slave log file.
CA_SLAVE_PRINT_MASK CA_MASTER_PRINT_MASK	These variables control which logging messages to ignore. They are code set within the code based on the values of the `LOG_` configuration file keys and should not be set directly.
CA_TRANSIT_INITIAL_WAIT	`Transit Initial Wait` specifies the number of timesteps a transit vehicle must be present at a transit stop before any passengers get on or off.

Configuration File Key	Description
CA_USE_NETWORK_CACHE	If set, use a cached binary representation of the network. This representation would have been created by a prior run of the simulation.
CA_USE_PARTITIONED_ROUTE_FILES	It is more efficient for slaves to read only those plans that start in the part of the network for which they are responsible. If the partitioning to be used by the simulation is available (for example, from a prior run of the simulation), the *DistributePlans* utility will create a separate pair of indexes for each slave into one common plan file. If `Use Partitioned Route Files` is set, the slaves will look for these slave-specific indexes. If they do not exist, the simulation will fall back to using a single global pair of indexes.
CA_USE_ROMIO_FOR_OUTPUT	If `Use Romio For Output` is set, and the executable was compiled with the `USE_ROMIO` and `USE_MPI` flags defined, the parallel output system will use ROMIO files instead of Unix files.
PAR_HOST_COUNT	The number of distinct machines that make up the parallel machine environment.
PAR_HOST_I PAR_HOST_CPUS_I PAR_HOST_SPEED_I	These variables describe the parallel machine environment to the simulation. There should be one set of these three variables, with I replaced by an integer from 0 to the value of `PAR_HOST_COUNT` – `1`, for each host. Host should be a string containing the name of the machine. `Host CPUs` should give the number of CPUs available for use on the machine. `Host Speed` should give the relative speeds of the different machines in arbitrary units. The sum of all the values of `Host CPUs` must be at least one larger than the number of slaves requested.
PAR_MIN_CELLS_TO_SPLIT	The minimum number of cells that a link must have in order to be split between CPU nodes. The partitioning software will not split links with fewer cells than this number. Recommended value is 10. This key is required.
PAR_PARTITION_FILE	The full pathname of the file where partitioning information will be saved. If this file exists, the partition will be read from the file. If `PAR_SAVE_PARTITION` is set to a non-zero value or `PAR_USE_METIS_PARTITION` and `PAR_USE_OB_PARTITION` are not set, this key is required.
PAR_RTM_INPUT_FILE RTM_FEEDBACK_FILE RTM_SAMPLE_INTERVAL PAR_RTM_PENALTY_FACTOR	The partitioning algorithms try to find the partition that spreads the computation associated with nodes and links evenly while simultaneously trying to minimize the communication costs associated with split links. The costs for each node and link can be estimated using run time costs from prior runs. These costs are sampled at the interval defined by RTM Sampling Interval and written out to the file named by RTM File. They are read in from the file found in the directory named by `OUTPUT_DIRECTORY`.

Configuration File Key	Description
`PAR_SAVE_PARTITION`	If set to a non-zero value, the network partition will be saved in the file specified by the configuration file key `PAR_PARTITION_FILE`. This key is required.
`PAR_SLAVES`	This key sets the number of slave processes to spawn. It must be smaller than the number of host CPUs available (to allow one process for the master).
`PAR_USE_METIS_PARTITION`	If set to 1, the METIS algorithm will be used to partition the transportation network among the slave processes. Behavior is undefined if both `PAR_USE_METIS_PARTITION` and `PAR_USE_OB_PARTITION` are set to non-zero values. If neither is set, the Traffic Microsimulator will attempt to read a partition file specified by the `PAR_PARTITION_FILE` configuration file key and will exit with an error if the file is not readable.
`PAR_USE_OB_PARTITION`	If set to 1, an orthogonal bisection algorithm will be used to partition the transportation network among the slave processes. Behavior is undefined if both `PAR_USE_METIS_PARTITION` and `PAR_USE_OB_PARTITION` are set to non-zero values. If neither is set, the Traffic Microsimulator will attempt to read a partition file specified by the `PAR_PARTITION_FILE` configuration file key and will exit with an error if the file is not readable.
`PLAN_FILE`	The plan file specifies the name of the file in which plans reside or a string to which *.tim.idx* and *.trv.idx* can be appended to find the time-sorted and traveler-id-sorted indexes into a plan file(s). The plans should include all travelers; for example, plans created by the Route Planner, transit driver plans, freight plans, etc. The name should be given as an absolute pathname because the slave executables are not always run from the current working directory.
`VEHICLE_FILE`	The vehicle file specifies the name in which vehicles reside or a string to which *.veh.idx* can be appended to find the vehicle-id-sorted index into a vehicle file(s). The vehicle file must include all vehicles to be used in the simulation.
`VEHICLE_PROTOTYPE_FILE`	The vehicle prototype file must include information about every vehicle type used in the simulation.

Appendix B: Configuration File Keys for Snapshot Output

Configuration File Key	Description
OUT_SNAPSHOT_BEGIN_TIME_n	The first time (in seconds from the midnight before simulation start) at which to collect data.
OUT_SNAPSHOT_END_TIME_n	The last time (in seconds from the midnight before simulation start) at which to collect data.
OUT_SNAPSHOT_FILTER_n	The list of expressions (where each expression has the form FIELD OPERATOR VALUE and multiple expressions are separated by semicolons) for filtering records. Valid values for FIELD are found in Tables 3-5, and values for OPERATOR are found in Table 12.
OUT_SNAPSHOT_LINKS_n	The path of the link specification (described in Table 11).
OUT_SNAPSHOT_NAME_n	The file name for snapshot output.
OUT_SNAPSHOT_NODES_n	The path of the node specification (described in Table 10).
OUT_SNAPSHOT_SUPPRESS_n	The list of fields (separated by semicolons) not to include in the output file.
OUT_SNAPSHOT_TIME_STEP_n	The frequency (in seconds) at which to report data (i.e., write it to disk).
OUT_SNAPSHOT_TYPE_n	The types of snapshot output to collect (separated by semicolons) permissible values are VEHICLE; INTERSECTION; SIGNAL.

Appendix C: Configuration File Keys for Event Output

Configuration File Key	Description
`OUT_EVENT_BEGIN_TIME_n`	The first time (in seconds from the midnight before simulation start) at which to collect data.
`OUT_EVENT_END_TIME_n`	The last time (in seconds from the midnight before simulation start) at which to collect data.
`OUT_EVENT_FILTER_n`	The list of expressions (where each expression has the form `FIELD OPERATOR VALUE` and multiple expressions are separated by semicolons) for filtering records. Valid values for `FIELD` are found in Table 2, and values for `OPERATOR` are found in Table 12. Valid values for `VALUE` must be expressed in decimal notation (not hexadecimal).
`OUT_EVENT_NAME_n`	The file name for event output.
`OUT_EVENT_SUPPRESS_n`	The list of fields (separated by semicolons) not to include in the output file.
`OUT_EVENT_TYPE_n`	The types of event output to collect permissible value is `TRAVELER.`

Appendix D: Configuration File Keys for Summary Output

Configuration File Key	Description
OUT_SUMMARY_BEGIN_TIME_n	The first time (in seconds from the midnight before simulation start) at which to collect data.
OUT_SUMMARY_BOX_LENGTH_n	The length of the boxes (in meters).
OUT_SUMMARY_END_TIME_n	The last time (in seconds from the midnight before simulation start) at which to collect data.
OUT_SUMMARY_ENERGY_BINS_n	The number of bins used to cover the range of the energy histogram.
OUT_SUMMARY_ENERGY_MAX_n	The maximum energy in the energy histogram.
OUT_SUMMARY_ENERGY_SOAK_n	The single value specifying the soak time for which to collect energy data. Permissible values are NEGLIGIBLE; SHORT; MEDIUM; or LONG. If a key is not specified, all soak times are included in the energy output.
OUT_SUMMARY_FILTER_n	The list of expressions (where each expression has the form FIELD OPERATOR VALUE and multiple expressions are separated by semicolons) for filtering records. Valid values for FIELD are found in Tables 6-9, and values for OPERATOR are found in Table 12.
OUT_SUMMARY_LINKS_n	The path of the link specification file (described in Table 11).
OUT_SUMMARY_NAME_n	The file name for summary output.
OUT_SUMMARY_SAMPLE_TIME_n	The frequency (in seconds) at which to accumulate data.
OUT_SUMMARY_SUPPRESS_n	The list of fields (separated by semicolons) not to include in the output file.
OUT_SUMMARY_TIME_STEP_n	The frequency (in seconds) at which to report data (i.e., write it to disk).
OUT_SUMMARY_TYPE_n	The types of summary output to collect (separated by semicolons) permissible values are DENSITY; TIME; VELOCITY;or ENERGY.
OUT_SUMMARY_VEHICLE_TYPE_n	The vehicle type and subtype (separated by colon) for which to collect velocity data. If subtype is zero or not specified, data for all subtypes of type will be included in the velocity output. If key is not specified, all vehicle types will be included in the velocity output.
OUT_SUMMARY_VELOCITY_BINS_n	The number of bins used to cover the range of the velocity histogram (in meters/second).
OUT_SUMMARY_VELOCITY_MAX_n	The maximum velocity in the velocity histogram (in meters/second).

Appendix E: Default Output Configuration File Keys

Configuration File Key	Description
`OUT_BEGIN_TIME_DEFAULT`	The first time (in seconds from the midnight before simulation start) at which to collect data.
`OUT_EASTING_MAX_DEFAULT`	The maximum easting (in meters) for which to report data (currently unused).
`OUT_EASTING_MIN_DEFAULT`	The minimum easting (in meters) for which to report data (currently unused).
`OUT_END_TIME_DEFAULT`	The last time (in seconds from the midnight before simulation start) at which to collect data.
`OUT_EVENT_FILTER_DEFAULT`	The list of expressions (of the form `FIELD; OPERATOR; VALUE;` — separated by semicolons) for filtering event records.
`OUT_EVENT_SUPPRESS_DEFAULT`	The list of fields (separated by semicolons) not to include in the event output file.
`OUT_LINKS_DEFAULT`	The path of the link specification file.
`OUT_NODES_DEFAULT`	The path of the node specification file.
`OUT_NORTHING_MAX_DEFAULTY`	The maximum northing (in meters) for which to report data (currently unused).
`OUT_NORTHING_MIN_DEFAULTY`	The minimum northing (in meters) for which to report data (currently unused).
`OUT_SNAPSHOT_FILTER_DEFAULT`	The list of expressions (of the form `FIELD; OPERATOR; VALUE;` — separated by semicolons) for filtering snapshot records.
`OUT_SNAPSHOT_SUPPRESS_DEFAULT`	The list of fields (separated by semicolons) not to include in the snapshot output file.
`OUT_SNAPSHOT_TIME_STEP_DEFAULT`	The frequency (in seconds) at which to report snapshot data (i.e., write it to disk).
`OUT_SUMMARY_BOX_LENGTH_DEFAULT`	The length of the summary data boxes (in meters).
`OUT_SUMMARY_ENERGY_BINS_DEFAULT`	The number of bins used to cover the range of the energy summary histogram.
`OUT_SUMMARY_ENERGY_MAX_DEFAULT`	The maximum energy in the energy histogram (in cells-squared per second-squared).
`OUT_SUMMARY_FILTER_DEFAULT`	The list of expressions (of the form `FIELD; OPERATOR; VALUE;` — separated by semicolons) for filtering summary records.
`OUT_SUMMARY_SAMPLE_TIME_DEFAULT`	The frequency (in seconds) at which to accumulate summary data.
`OUT_SUMMARY_SUPPRESS_DEFAULT`	The list of fields (separated by semicolons) not to include in the summary output file.
`OUT_SUMMARY_TIME_STEP_DEFAULT`	The frequency (in seconds) at which to report summary data (i.e., write it to disk).
`OUT_SUMMARY_VELOCITY_BINS_DEFAULT`	The number of bins used to cover the range of the velocity summary histogram.
`OUT_SUMMARY_VELOCITY_MAX_DEFAULT`	The maximum velocity in the velocity histogram (in meters per second).

Appendix F: CA Error Codes

Error codes for the CA are in the range 13000 – 13999.

Code	Description
13003	Not used.
13004	Not used.
13005	Not used.
13006	The slave or master cannot open output files or find required configuration file keys.
13007	The master cannot partition the network into pieces for each slave to handle, or the pieces cannot be broadcast to the individual slaves.
13008	The master or slave cannot find and open all of the required network data files, or the cached network data files are corrupted.
13009	A slave does not acknowledge having read the plans that are required before the first simulation step.
13010	The master cannot send a message to each slave directing it to execute a sequence of timesteps.
13011	The master cannot send a message to each slave directing it to execute a single timestep.
13012	Not used.
13013	The first timestep was not completed.
13014	The parallel communication system broke down.
13015	The master cannot shut down the simulation system.
13016	A slave did not correctly flush its output buffers when it ended.
13017	The master cannot flush run time statistics into the run time monitor output file.
13018	The `PLAN_FILE` configuration file key is missing from the configuration file.
13019	The file specified by the `PLAN_FILE` configuration file key cannot be opened for reading.
13020	The `VEHICLE_FILE` configuration file key is missing from the configuration file.
13021	The `VEHICLE_PROTOTYPE_FILE` configuration file key is missing from the configuration file.
13022	The `OUT_DIRECTORY` configuration file key is missing from the configuration file.
13023	The `CA_USE_NETWORK_CACHE` configuration file key is turned on but the cache files are not readable.
13024	An error in the METIS or orthogonal bisection routines prevents partitioning of a network.
13025	A saved network partition file specified by the `PAR_PARTITION_FILE` configuration file key cannot be used because it assumes a different number of slaves than specified by the `PAR_SLAVES` configuration file key.
13026	The list of machines or the location of the current CPU in that list could not be determined.
13027	A slave could not correctly flush output from a timestep or read in any plans required for executing the next timestep.
13028	A slave could not find information (e.g., a name) for the host it resides on.
13029	A slave is building its portion of the network if it cannot identify the CPU containing the node at the far end of a distributed link.
13030	A slave received a corrupted message with an unknown event ID.

Code	Description
13031	The vehicle or related index specified by the `VEHICLE_FILE` configuration key cannot be processed.
13032	There is a problem with the transit stop data.
13033	The master does not receive expected acknowledgments from all slaves after prompting them to take an action such as executing a timestep or exchanging boundary information.
13034	Unexpected messages were received while a slave was waiting for boundary information from its neighbor CPUs.
13035	A slave received information about a distributed link that is not a part of the portion of network on that slave during a boundary information exchange.
13036	A slave cannot allocate memory for a message during a boundary information exchange.
13037	A slave cannot send a message to a neighboring CPU during a boundary information exchange.
13038	A vehicle was removed from the simulation while it had occupants.”
13039	A capacity constraint on a vehicle was violated.
13040	The simulation attempted to remove the vehicle used internally to represent a blocked cell in a lane.
13041	A driver attempted to enter a vehicle that already had a driver, or the first node in a driver’s plan could not be reached from the origin parking lot.
13042	The source of an error cannot be determined.

Appendix G: Output Representation Error Codes

Error codes for the Output representation are in the range 20000 – 20999.

Code	Description
20001	Caught signal.
20002	Assertion failed.
20003	Invalid program arguments.
20004	Standard exception.
20005	Unknown exception.
20006	Output subsystem problem.
20007	Storage failure.
20008	Writer failure.
20009	Invalid processor.
20010	Cannot read.
20011	Input error

Chapter Five: Index